1995年7月1日発行（毎月1回1日発行）第14巻7号通巻159号 昭和58年11月2日第三種郵便物認可

PERSONAL COMPUTER MAGAZINE for MZ, X1, and X68000

特集 Optimizing Method

一線を超えた68系プログラマ養成講座/GCCにおける最適化/
浮動小数点演算プロセッサの効果/Xellent30を活用する

THE USER'S WORKS SPECIAL/新製品紹介PDドライブLF-1000

1995

SOFT BANK オー／エックス
定価680円

SHARP

目の付けどころが、
シャープでしょ

フィールドが、膨らむ。

68買ったらEXEクラブへ入ろう!

先が、ますます面白くなる。

●

未来への確かなビジョンをベースに
発展性のあるプラットホームとしてのウィンドウ環境を提供する
国産オリジナルウィンドウシステムSX-WINDOW。

●

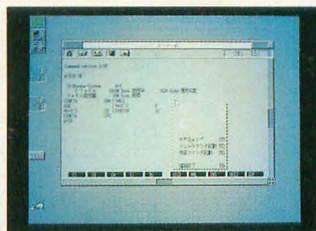
GUI環境や操作環境、高速化へのゆるぎない探求、
マルチメディアの統合的なハンドリング。

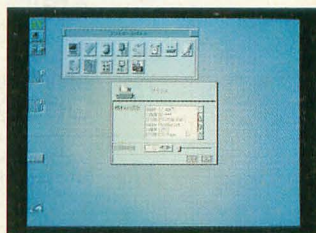
●

いま、より多彩なフィールドへ
そのインテリジェンスが展開を始める。

●

次のステージが見えてくる。

●インライン入力のサポート：ASK68K Ver.3.0を利用したインライン入力をSX-WINDOWで実行可能。またシャーペン.Xをワープロとして利用できるよう、さまざまな機能が付加されています。

●コンソールをサポート：Human68kやX-BASICのコマンドをSX-WINDOWアプリケーションと同時にタイムシェアリングで実行できます。
（グラフィックを利用したものなど、SX-WINDOWと処理が重複するものは実行できません。）

SX-WINDOW ver.3.1の
データ利用環境

●多彩なプリンタに対応：さまざまなSX-WINDOWアプリケーションで利用できるページプリンタドライバを標準装備。ESC/Page、LIPS III、PostScriptに対応したプリンタが利用できます。

今も、先も楽しめる。
いつも新展開の予感、SX-WINDOWのニューバージョン。

SX-WINDOW ver.3.1

「SX-WINDOW ver3.1システムキット」CZ-296SS（130㎜FD）/CZ-296SSC（90㎜FD） 標準価格22,800円（税別）

EXEクラブって何だ?

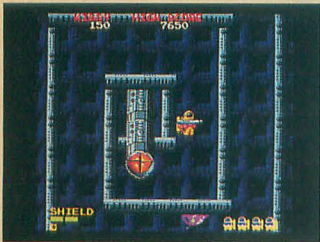
X68030/X68000を手に入れて、いろいろチャレンジしたい皆さん。情報のチャンネルは多いほどいいですよね。ということで、EXEクラブは68ユーザーのための水先案内人。あなたのチャレンジを強力にバックアップしますよ。

本体同梱の入会申込ハガキを送るだけで、自動的に無料入会。さらに下記の特典付き。

Oh!X

CONT

〈スタッフ〉
●編集長／前田 徹 ●副編集長／植木章夫 ●編集／山田純二 高橋恒行 ●協力／有田隆也 中森 章 林 一樹 吉田幸一 華門真人 朝倉祐二 大和 哲 村田敏幸 丹 明彦 三沢和彦 長沢淳博 清瀬栄介 柴田 淳 瀧 康史 横内威至 進藤慶到 菊地 功 伊藤雅彦 ●カメラ／杉山和美 ●イラスト／山田晴久 江口響子 高橋哲史 川原由唯 ●アートディレクター／島村勝頼 ●レイアウト／元木昌子 加藤真二 ●校正／グループごじら

特集 Optimizing Method

パラデューク

THE USER'S WORKS SPECIAL

ビジネスショウ'95

PDドライブLF-1000

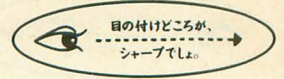
ショートプロぱーてぃ

表紙絵：塚田　哲也

E　N　T　S

1995 JUL. 7

■広告目次

ビデオグラフィックスの
世界へ。
■お問い合わせは…シャープ株式会社
電子機器事業本部システム機器営業部 〒545 大阪市阿倍野区長池町22番22号 ☎(06)621-1221(大代表)
資料請求券
X68030

1,677万色対応、ビデオ映像を高画質・高速取り込み

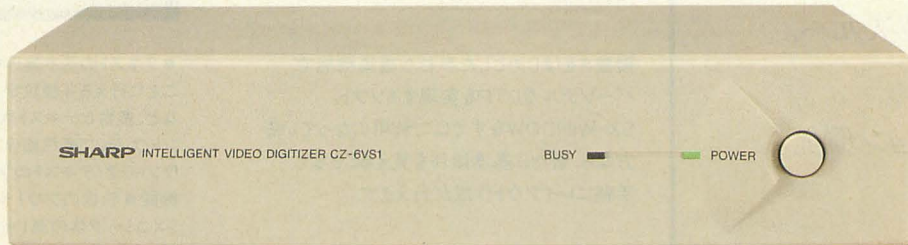
テレビやビデオ、ビデオディスクなどの映像をX68シリーズやMacシリーズ※1の動画・静止画データとして高速取り込みが可能、いわば"ビデオスキャナ"とでも呼びたいビデオ入力ユニットです。1,677万色対応、最大640×480ドットの高解像度※2。動画・静止画の手軽なハンドリングが、新たなグラフィックシーンを創造します。

※1 MacintoshはIIシリーズ以降の機種に対応、ディスプレイ解像度が640×480ドットの場合、取り込み可能な範囲は、160×120ドット、320×240ドットのサイズになります。
※2 X68030/X68000シリーズでは、1,677万色はデータ作成のみに対応。表示は最大65,536色、解像度は512×512ドット。また、Macintoshは機種により表示色数が異なります。

アプリケーションツール「ライブスキャン」を標準装備

動画や静止画を簡単に保存できるアプリケーションソフト「ライブスキャン」※を標準装備。取り込んでいる映像を表示したり、残したいシーンを簡単に静止画保存したり、手軽な動画・静止画ハンドリングでパソコンの可能性をさらに広げます。X68030/X68000シリーズ用SX-WINDOW対応版とMacintoshシリーズ用QuickTime対応版の2種類を同梱しています。

※SX-WINDOW版はバージョン3.0以降（メモリー4MB以上）、QuickTime版はMacintosh漢字Talk7リリース7.1以上のシステムとQuickTime1.5以上（メモリー8MB以上）が必要です。

1,677万色対応の高速映像取り込み、動画・静止画の手軽なハンドリングが、新たなマルチメディアシーンを創造する。

■SCSIインターフェイス採用：パソコンの専用I/Oスロットを使わずに接続可能になり、汎用化を実現しました。またSCSI-2（FAST）インターフェイスの採用により、データ転送速度の高速化を図っています。X68030/X68000シリーズでは、SCSI-2（FAST）対応のハードディスクを接続することにより、パソコン本体を経由しないで、ハードディスクに直接、動画データをテンポラリデータとして記録することが可能です。パソコン本体のハードディスクへは、記録終了後に、テンポラリデータを変換し動画データとして保存できます。

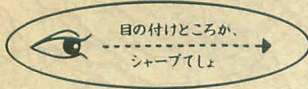
※CZ-600C/601C/611C/602C/612C/652C/662C/603C/613C/653C/663Cに接続する場合は別売のSCSIインターフェイスボードCZ-6BS1ならびにSCSI変換ケーブルCZ-6CS1が必要です。※CZ-604C/623C/634C/644Cに接続する場合は、別売のSCSI変換ケーブルCZ-6CS1が必要です。
※Macintosh Power Bookシリーズに接続する場合は別売のSCSIケーブルなどが必要です。詳しくはMacintosh Power Bookシリーズの取扱説明書をご覧ください。

■高機能MPUを搭載：クロック周波数25MHzの32ビットMPU/MC68EC020を搭載、高速処理やパソコン本体の負担の軽減を実現します。

●MacはMacintoshの略称です。●Macintosh、Macintosh IIは、米国アップルコンピュータ社の登録商標です。●Power Bookは米国アップルコンピュータ社の商標です。●漢字Talk7はアップルコンピュータジャパン社の商標です。●QuickTimeは、米国アップルコンピュータ社の商標です。●価格には、消費税及び配送・設置・付帯工事費、使用済み商品の引き取り費等は含まれておりません。

ビデオ入力ユニット

CZ-6VS1

標準価格178,000円（税別）

For X68030/X68000series

ORIGINAL SOFTWARE COLLECTION

さらに高度な創造次元へ。
ますます成熟する
そのアプリケーション環境。

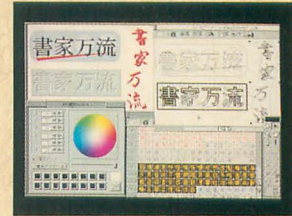
NEW アプリケーション

●独自のアウトラインフォントを付属

フォント&ロゴ デザインツール 書家万流 SX-68K

CZ-282BWD　標準価格29,800円(税別)

4MB | ver.3.0 | HD 10MB

フォントやロゴを手軽に作成するための
デザインツール。作成したロゴは
クリップボードを介し、シャーペンや
EGWord SX-68K、XDTP SX-68Kなど
他のアプリケーションで利用できます。

●SX明朝体/SXゴシック体フォント(JIS第1水準&第2水準)を付属●ベジェ曲線のアウトライン編集によるデータ作成●フォントファイル全体にわたってのエフェクト処理●既存のフォントファイルからのデータ抽出、ドローオブジェクトへのエフェクト処理●複数のフォントファイルをリンクして新たなフォントファイルの作成が可能●65,536色表示で確認しながらロゴ作成ができるグラフィックウィンドウ(GRW.X)対応

●パーソナルDTPをX68で

XDTP SX-68K

CZ-291BWD　標準価格35,000円(税別)

4MB | ver.3.0 | HD 5MB

縦書きをはじめとした多彩な編集機能で
パーソナルなDTPを実現するソフト。
SX-WINDOWをすでにご利用になっている
方なら、新たに基本操作を覚えることなく
手軽にレイアウト作成が行えます。

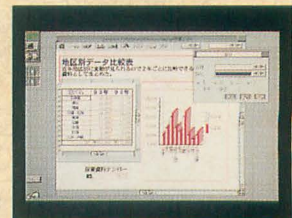
●テキストの基本処理をはじめ、テキストフレームごとに行える各種設定、スタイル別の検索/置換など、豊富なテキスト編集機能●グラフィックウィンドウ、そして各種画像フォーマットへの対応●グラフィック/テキストのフレームから独立した罫線機能●独自のアウトラインフォント(SX明朝体、SXゴシック体の第1水準)標準添付●ページの移動/作成/削除がスピーディに行える独立したページウィンドウをサポート

●DTP感覚で自在にレイアウト編集

Datacalc SX-68K

CZ-273BWD　標準価格59,800円(税別)

4MB | ver.3.0 | HD 3MB

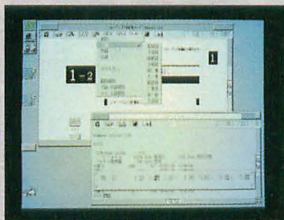
SX-WINDOW対応の新世代統合ソフト。
表計算、グラフ、データベース、テキスト、罫線の
各データを1枚の用紙に重ね合わせ、
移動、サイズ変更など
DTP感覚でレイアウト編集ができます。

●カルクシートでは、セル番地を意識することのない直感的なセル指定が可能●データベースフィールドでは、同一項目でもデータ型/データ長の異なったデータを管理できるなど、自由な設計が特長●データベースフィールドで入力したデータをカルクシートのデータとして利用したり、カルクシートのデータ変更を自動的にグラフ表示に反映させたり、同一データからさまざまな分析が可能なデータリンクもサポート

システム & アプリケーション

●さらに実用的なウィンドウシステムへの進化

SX-WINDOW ver3.1 システムキット

CZ-296SS(130㎜FD)/CZ-296SSC(90㎜FD)　標準価格22,800円(税別)　4MB

ASK68K ver.3.0を利用したインライン入力のサポート、Human68k/BASICコマンドをSX-WINDOWアプリケーションと同時にタイムシェアリングで実行できるコンソールのサポートをはじめ、シャーペン.Xをワープロとして利用できるよう機能アップ。また、さまざまなSX-WINDOWアプリケーションで利用できるページプリンタドライバを標準装備。ドローデータ(FSX)/フォントデータ(IFM)処理の高速化も実現しています。

※コンソールでは、SX-WINDOWと処理が重複するものは実行できません。

●SX-WINDOWを楽しく使うためのアクセサリ集

SX-WINDOW デスクアクセサリ集

CZ-290TWD　標準価格14,800円(税別)

SX-WINDOWをさらに便利に、楽しく使うためのデスクアクセサリ集です。スクリーンセーバ、スクラップブック、アドレス帳、電子手帳通信ツールなど、12種の豊富なアクセサリが収められています。

4MB | ver.3.0

●SX-WINDOW対応ドローイングツール

Easydraw SX-68K

CZ-264GWD　標準価格19,800円(税別)　4MB | ver.3.0

イラスト、フローチャート、地図、見取り図など各種グラフィックが製図感覚で作成できます。作成したデータは他のSX-WINDOW対応アプリケーションでも利用でき、企画書などの作成をサポートします。

●ウィンドウ対応のグラフィックツール

Easypaint SX-68K

CZ-263GWD　標準価格12,800円(税別)　2MB | ver.1.1

マウスによる簡単操作、65,536色中16色の多彩な表現、クリエイティブマインドに応えるウィンドウ対応のペイントツールです。同時に複数のウィンドウを開いて編集でき、各ウィンドウ間のデータ交換も行えます。

●定評のGUI対応ウィンドウワープロ

EGWord SX-68K

CZ-271BWD　標準価格59,800円(税別)

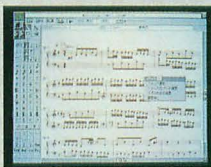
キャラクタベースのワープロを超えたGUIによる、手軽なDTPソフトとしても優れた表現力を発揮。定評ある日本語入力方式によるインライン入力、各種グラフィックデータやテキストデータの貼り込みができます。

4MB | ver.2.0 | HD 5MB

●グラフィック感覚の楽譜入力をサポート

MUSIC SX-68K

CZ-274MWD　標準価格38,000円(税別)

MIDI、FM、ADPCMに対応した楽譜ワープロ&作曲演奏ソフト。自由なレイアウトで、グラフィックを描くように楽譜入力。全パートの同時入力・編集、自動伴奏機能、多彩なプリンタ対応で美しい印刷も行えます。

4MB | ver.3.0

●マルチタスク機能をはじめ通信環境がさらに充実

Communication SX-68K

CZ-272CWD　標準価格19,800円(税別)

通信環境をさらに高めたウィンドウ対応の通信ソフト。マルチタスク機能により他のアプリケーションを実行中でも簡単に通信が可能。自動ログイン機能やプログラム機能など、豊富な機能をサポートしています。

2MB | ver.1.1

開発支援ツール

●X68030/X68000対応開発ツール

NEW KIT

CZ-295LSD　標準価格44,800円(税別)

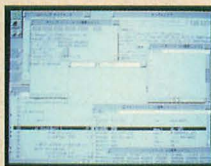
C compiler PRO-68KのX68030/X68000対応版。従来からの機能に加えて、Human68k ver.3.0、ASK68K ver.3.0にも対応。新たにGPIBライブラリ、MC68882対応フロートライブラリを付属しています。

2MB

●SX-WINDOWソフト開発支援ツール

SX-WINDOW 開発キット Workroom SX-68K

CZ-288LWD　標準価格39,800円(税別)

SX-WINDOW用のソフトウェア開発に必要なツールや33種類のサンプルプログラムを装備。プログラムの編集、リソースの作成、コンパイル、デバッグといった一連の作業がきわめて効率よく実行できます。

※ご使用に当ってはC compiler PRO-68K ver.2.1が必要です。

4MB | ver.2.0

●SX-WINDOW開発キットのサポートツール

開発キット用ツール集

CZ-289TWD　標準価格12,800円(税別)

「SX-WINDOW開発キット」をさらに使いやすくするためのサポートツール集。SXコールの簡易リファレンスを収めたインサイドSX、イベントハンドラ、ヒープビューアなど11種類のツールが用意されています。

4MB | ver.2.0

4MB | ver.3.0 | HD 10MB の表示は、メインメモリ4MB以上、SX-WINDOW ver.3.0以上、10MB以上の空きのあるハードディスクが必要であることを示しています。●EGWordは株式会社エルゴソフトの登録商標です。

●お問い合わせは…シャープ㈱電子機器事業本部（液映）システム機器推進プロジェクトチーム　〒162 東京都新宿区市谷八幡町8番地 ☎(03)3260-1161(大代表)へ　シャープ株式会社

資料請求券 CZソフト Oh!X 7係

SHARP

高速、高解像度。

透過原稿・ADF対応型カラーイメージスキャナ、誕生。

［第50回］
創造力というツール

江口　響子

響子inCGわ〜るど

創造力というのも道具……ツールにすぎないんだよ。T先生は，ぽつりとおっしゃった。とある専門学校の入学式で，初めてお会いしたときのことである。

T先生は美術教育に半生を費やした方だった。そして，70歳をすぎてつぶやいた言葉である。

まだ，その折り返し地点にも達していない私が，意味を完全に理解するのは無理だろう。しかし，T先生のおっしゃった言葉は，ずっと心の片隅で響いている。

＊

絵を描いたり，作曲したりという行為は，人が自分の創造力を使って成し遂げることだ。それが，コンピュータの出現で大きく変わってきている。

コンピュータでなにかを創造とする行為をとらえるには，だいたい2とおりのアプローチがあるようだ。

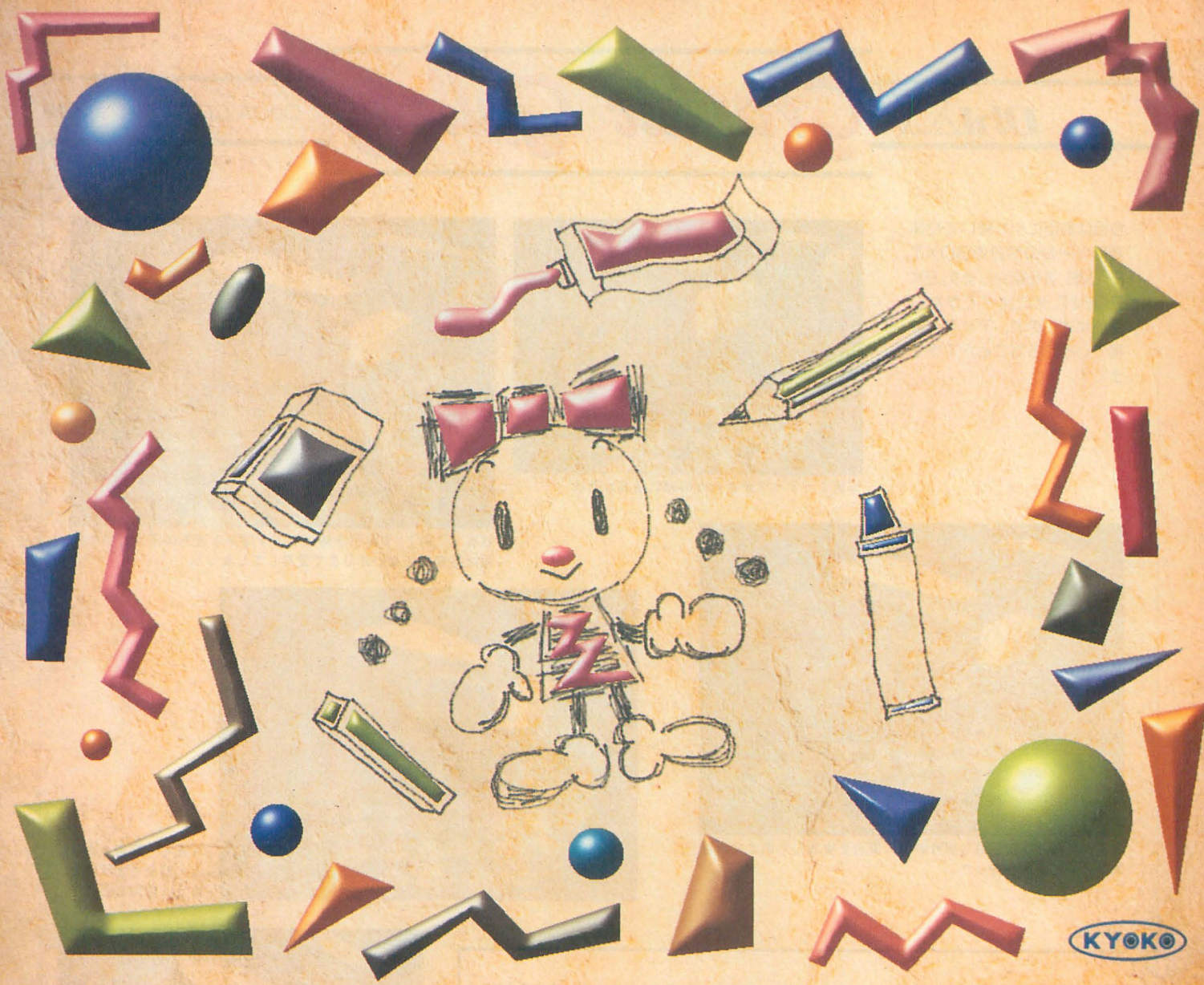
ひとつは，コンピュータによる創造そのものの自動化。CGでいうならば，プログラミングによって，オートマティックにコンピュータが画像を描き出すのがそうだ。ひとたび動き出してしまえば，人間の思惟が介入することはない。

もうひとつは，創造のツールとしてのコンピュータ。これは，創造を行う人の存在が前提である。その人は，持っているイメージをコンピュータによって取り出し，増幅させたり変形させたりする。グラフィックのアプリケーションを用いて，自分の作りたいものを速く，あるいは美しく描き出そうとするのは，この「CGわ〜るど」で一貫してきたことである。

ただ，実際の制作では両方のアプローチが混在することが多い。たとえば，MATIERによる自動描画機能を用いて背景を作ったのち，納得のいくまでモデリングした3Dのオブジェクトを配置する……フラクタル図形を描かせてから，それを部屋の壁にマッピングデータとして使う……という具合にだ。

創造という行為を，コンピュータによって自動化するのは人類の長い歴史のなかで画期的なことに違いない。そして，そのスタンスこそが，コンピュータによる創造の真の意味だ，と主張する人は大勢いる。

が，その考え方だけに固執するのは，私にはやや抵抗がある。

創造とは，新たな価値を実現しようとする活動そのものをいう。自分が楽しんだり，考えたり，

またその体験を多者と共有するのにつながる行為だ。ざっくばらんに，いろいろなアプローチがあってもいいのではないかと思う。

＊

ところで「なぜ，X68000にこだわり続けるのか」と聞かれることがある。マシンスペックとコストパフォーマンスから見ても，優れたマシンがどんどん出てきているのに……。

その問いには，いつも私はこう答えている。コンピュータでものを作ることについて考えさせてくれた原点のマシンが，ほかならぬこのX68000で，まあ，ツールというより相棒みたいなものですから……と。

今回のCGデータ

1280×1024ピクセル
1670万色フルカラーを4×5ポジで出力
作成手順
背景はフルカラーの取り込み画像。MATIERの立体ペイントと，ぼかしを加えたラインツールで作成したのちRGB更新セーブで保存。

P.S. おかげさまでCGわ〜るども50回を迎えました。読み続けてくださった皆さん，よい機会を与えてくださったOh!X編集部の方々にとても感謝しています。

今回は前回に引き続いて物体変形ツールの第2弾BOXTRANS.Xを紹介します。EXPOINT.Xに比べてかなり使いやすいので，アイデア次第でいろいろなことができそうです。

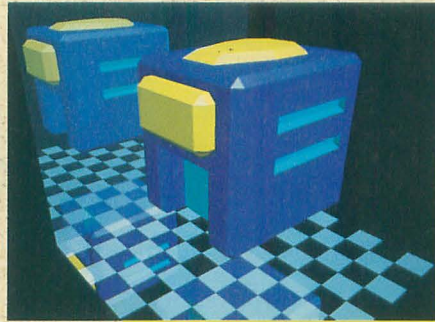



写真1　すべてがどれかの軸に垂直な形状

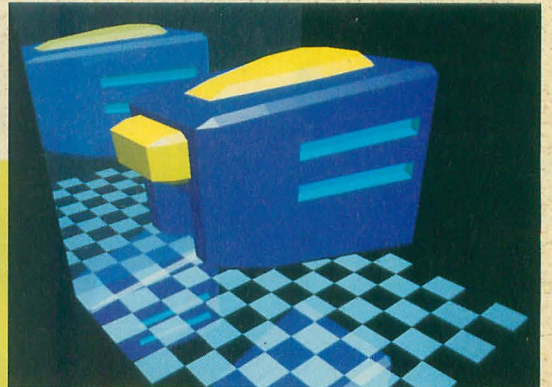



写真2　前2/3を引き伸ばしてすぼめる変形の予定だったが，領域指定してない部分まで変形してしまった

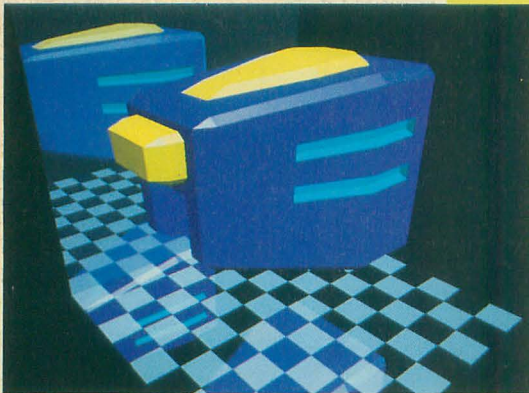



写真3　領域指定される部分の境に頂点を生成したあとで，写真2と同じ変形を行った

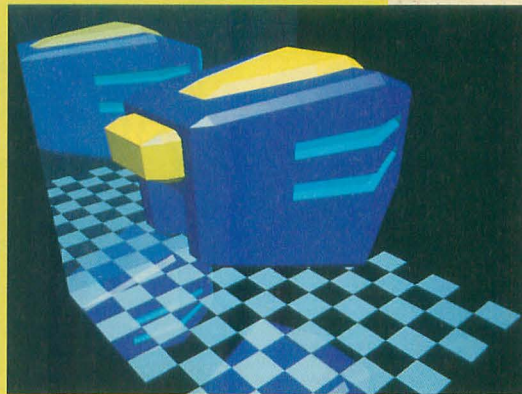



写真4　写真3の後ろの部分をすぼめるように変形した

写真5　写真3，4のような変形をより大胆に行った

BOXTRANS.Xを使ったサンプルアニメーション。RENCON.Xを使用するため作画には多少時間がかかります。また，頭部のプロペラは別パーツにして変形。

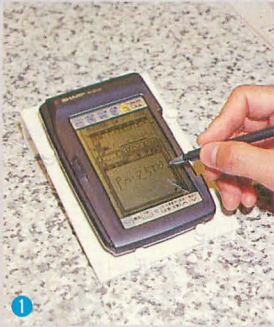
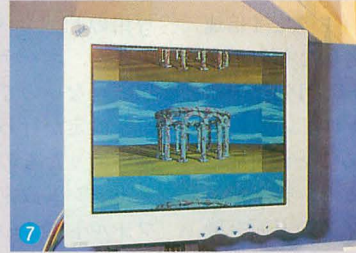
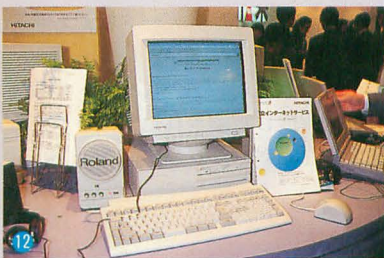
写真6－A　元の形状(プロペラは別)

写真6－B　頭部を大きく変形したもの

写真6－C　頭部を若干変形したもの

写真6－D　頭部の上部と後部の上部のローター接続部を2回に分けて変形したもの

ビジネスショウ'95

BUSINESS SHOW '95

5月17日～20日の4日間，東京晴海の国際見本市会場でビジネスショウ'95が開催された。全体的にPDA風の携帯機器などが目立ち，デジタルスチルカメラやペンコンピュータなども実用レベルの展開となってきたようだ。

意欲的に新技術を展開していたのはキヤノン，リコーなど。キヤノンではカラープリンタは当然として，FLC液晶ディスプレイ，PowerPC604を使ったパソコンの展示やRenderware1.4のマルチプラットフォームでの展示などが見られた。

FLC以外にも，液晶ディスプレイではIBMのTFT液晶ディスプレイが目を引いた。大型で実用的な解像度を持ち，そこそこの画質は出ていた。あとは値段次第か？

東芝ではデジタルビデオディスクを大々的にアピールしていたが，まだ実機に触れられる状態ではない。

そのほか，エレコムなどではZipドライブの展示も見られた。これは現在のFD技術の延長上にあるものだが，ドライブが2万円台と低価格，かつ高速であり，次世代の記憶メディアとして有望視されるものだ。

流行のインターネット関係の展示もいくつかあった。なかでも日立のインターネット体験コーナーで，いきなり「自分のホームページを作ろう」ってのはなかなか粋な試みではあったと思う。

❶シャープのWiz ❷FAXをFDに記録するFDファクス ❸書き換え可能3.5インチ光ディスク ❹シャープの21型TFTディスプレイ ❺DVDのデモ ❻各種PowerPCチップ ❼IBMのTFTディスプレイ ❽Renderware1.4のデモ ❾FLCディスプレイ ❿PowerPC604パソコン ⓫動画も撮れるスチルカメラ ⓬インターネット体験コーナー ⓭普通のテレビと電話をテレビ電話に ⓮音色エディットもできるWaveBlaster2 ⓯コピー紙を再利用できる「消えぞう君」 ⓰SC-55もカードサイズになった

PDドライブLF-1000

Nakano Shuichi 中野 修一

最近急に外部記憶装置の種類が増えてきた。そのうちのひとつがPDドライブである。すでにこれを内蔵したパソコンも発表されるなど，混迷する大容量リムーバブルメディアのなかで面白い位置を占めている。すでに市場にも出回り始めたようで，目にしたことのある人もいるかもしれない。

PDとは？ 相変化型光ディスクと4倍速CD-ROMドライブを一体化したものが今回紹介するPDドライブである。PDは一般的なSCSI機器として設定されているのでX68000シリーズでも使用することが可能だ。

記録原理を簡単に解説すると，書き込みレーザー（高出力）でメディアの表面を融点近くまで瞬間的に熱し，冷却速度を変えることで結晶質と非晶質の状態を作り出す。結晶状態の違いは読み込みレーザー（低出力）の反射率の違いとなって表れるので，それを読み取って信号とするわけだ。

このPDは650Mバイト(フォーマット時633Mバイト)の容量とちょっと前のハードディスククラスの速度を持っているが，現在の相変化型光ディスク（別途製品化されている）の技術レベルでいえばもっともっと性能を上げることは可能ということだ。コストやCD-ROMとの共用というあたりでメカ的な制限が入っているのだろう。

接続

製品にはPDモードとCD-ROMモードがある。

PDモードはSCSIでは光磁気ディスクとして扱われる。本当は「光ディスク」なので「磁気」は関係ないのだが，使い勝手はほぼ同じだ。X68030では特にデバイスドライバは必要としないが，X68000シリーズでは毎度のことながらSxSIやINQPATCHといったツールが必要になる。接続の手順は一般的なMOと同じと考えていい。

CD-ROMとして使用する場合には計測技研のCD-ROMドライバまたは同等品が必要だ。また，DIPスイッチ2番のLUNをONにしないとドライバが認識できないので必ず設定するように。

製品にはターミネータが内蔵されているので，SCSIの終端に置く場合は裏のDIPスイッチを入れるだけでよい。

PDモードとCD-ROMモードでは同じSCSI IDを使用するので，どちらかを使用中にはもう一方に切り替えることができない。リセット時にメディアが挿入されていたほうが使用されることになる。メディアを入れ替えてそのまま使用できるようにするには専用のドライバが必要になるだろう。

また，ユーティリティの類がまったく使用できないので，PD使用時のライトキャッシュ設定が行われない可能性がある。編集部で借りたものはすでにドライブにライトキャッシュ設定された状態だったので，デフォルトでこの設定になっているかどうかは不明。設定されていない場合（アクセスランプの色で判別できる），他機種につないで設定する必要がある（ライトキャッシュを使いたい場合にはだが）。

PDモード

まずはPDモードで使ってみよう。

容量が大きいこともあるが，フォーマットには30分弱かかる。

連続読み込みは秒間1Mバイト弱と結構速いのだが(ピーク性能はもっと高い)，普通のマシンではそれを実感できないだろう。ノーマルのX68030でもHSCSIなどを組み込まない限り本体側が追いつかない。X68000シリーズで使用する限りはハードディスク並みの転送速度といっても過言ではない。

ただし，ランダムアクセスはあまり速くない。ディレクトリ情報などは上手にキャッシングしてくれるので，操作のもたつきはほとんど感じないのだが，ファイルを大量に入れたディレクトリなどに移動すると若干の間があく。

ざっと使ってみた感じでは，ランダムアクセスはMO並み(か，やや劣る）で転送速度はハードディスク並み，という感触。平均シークタイム165msという値ほどには遅く感じないのはキャッシュが賢いからだろうか？

おかしい……。大きなメディアになると動作が重くなるのが通例（ヘッドの移動距離が大きい）なのであまり期待はしていなかったのだが，これならかなり快適に使用できる速度である。

どうやらディスクの外周から使用されていくようなので，使い始めの状態では間違いなく高速だが，使っているとだんだん遅くなることはありうる。理論上は最外周部の半分近くまで転送速度が落ちることになるが(3000回転の128MバイトMO程度の速度になる)，X68000で使う分にはさほど気にする必要はないはずだ。

じゃあ，と，今度はパーティションを切って最内周部だけを使うようにしてみた。重い……。異様に重い。たかが1Mバイトのファイル転送に7分弱かかっている？エラーセクタに引っかかっている可能性があるのでパーティションを切り直して試しても同様の結果。動作中ずっとシーク音が繰り返されている（外周部ではシーク音はほとんど聞こえない）。データ転送が小刻みだからだろうか？

fastioで連続転送量を128Kバイトに設定すると3分程度に短縮されたが，それでも内周部はかなり遅く，シーク回数も多い。

LF-1000 118,000円(税別) 松下電器産業

詳細は不明だが，ディスクの管理に多少疑問がある。参考までに設定は，

fastio -b1024 -p16 -s128 -w -f -d

である。とりあえず，大容量メディアではこういった対処が有効なことが多い（普通のMOでも）。

さて，PDのメリットはなんといっても容量の大きさだ。MOでは心許なかった大容量ハードディスクでも気軽にバックアップが取れる。MO何枚かを取っ替えてアニメーションデータを管理していたような人には朗報かもしれない（少数派だな……）。

完全にハードディスク代わりにするようなことはできないが，ランダムアクセスが遅いとはいっても，これだけの大きさのメディアに小さなファイルを山ほど詰め込んだり，頻繁にアクセスを必要とするものを入れるなどというのは常識的にやらないと思われる。あえていえばそれは使用法が間違っている。

実用上問題ないとはいえ，メディアの使用回数には理論上の限界があるので（一応，寿命は30年となっている）頻繁に書き換えを行うような用途には使うべきではないだろう（辞書を入れるとか，テンポラリに指定するとか）。

そのほか，ゾーンCAV方式のためか，連続領域でも急にアクセス速度が遅くなる点がいくつかあるので，一定の転送速度を期待するような用途には不向きといえるだろう（AMIの再生とか）。

CD-ROMモード

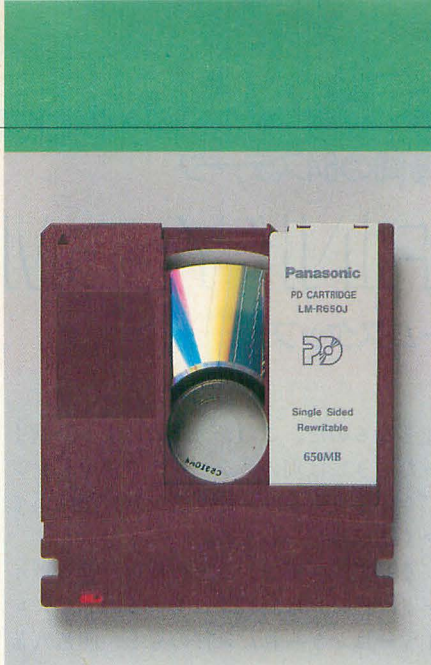
前述のとおり，LUNスイッチを設定してさえやれば特に問題はない。

オーディオトラックの演奏ができないということもないし，SX-WINDOWでもちゃんと動く。4倍速なのでストレスもたまらない。CD-ROMドライブにしては音質もまあまあいい。

欠点といえば音声トラックの取り込みができないということくらいだが，これはできないのが当たり前なので文句をいってもしょうがない。起動後にトレイを入れ直さないと認識してくれないのはほかのドライブと同じ。これはドライバの問題だろう。

CD-ROMとしてはまったく文句のつけようはない。

総括

問題は，共用型なので両者を同時に使えないということだ。同時に使うことがどれくらいあるかという問題が絡んでくるが，同時に使うことが少なければ共用型のメリットというのもある。2台に分かれていて倍のスペースや電源を確保しなければならないことや，SCSIケーブルを無駄に引き延ばすことを考えた場合だ。

そのほか，どちらの用途が多いかという問題も絡んでくるが，X68000ではたいていの人は「どちらもあまり使わない」というオチがついてしまうのがちょっと悲しい。

MOもCD-ROMも持ってないという人が買う場合はどうだろうか？

X68000ユーザーに限っていえば，MOの所有率は非常に高い。MIDIとMOについては他機種の平均をはるかに上回る普及率を示すという特殊な事情があり，大容量リムーバブルメディアのひとつの基準として存在しているので，こういった機器に手を出すのは128MタイプMOを入手してからにするのが無難だろう。

MOを持ったうえで，これだけの大容量メディアを使用する用途となると，大量のデータ整理を行う場合か，MOで間にあわない容量のハードディスクのバックアップ，ほかにはアニメーションの作成くらいしかないだろう。容量を生かせばもっと面白い用途もあるのだが，まだ時代が追いついていないというべきだろうか。

ドライブの価格はいまでもMO+CD-ROMと競合できるところだし，普及次第では今後の外部記憶装置の本命となる可能性も否定できない。しかし，普及すればするほど一体型の不利な点がクローズアップされてくることも明らかである。

今年の冬くらいには640MバイトタイプMOが発表されると思われるので，それからが次期記憶メディアレースの本番と見ていいだろう。面白い位置を突いてきたPDの健闘に期待しよう。

PDのメディア

■PD/CD-ROM ドライブ

項目			仕様
品番			LF-1000JA/JD
電源			AC100V 50/60Hz
消費電力			12W（シーク時以外10W）
対応インタフェイス			SCSI-2
シークタイム		PD	165ms
		CD-ROM	195ms
連続データ転送速度		PD	518KB/s～1,141KB/s
		CD-ROM	標　準150KB/s 4倍速600KB/s
データ転送速度（SCSI）		同　期	最大5.0MB/s
		非同期	最大3.3MB/s
ディスク回転数		PD	2,026min^{-1}（rpm）
		CD-ROM	標　準200min^{-1}（rpm）～530min^{-1}（rpm） 4倍速800min^{-1}（rpm）～2,120min^{-1}（rpm）
ディスク スタート/ストップ時間		PD	6s/3s
		CD-ROM	6s/3s
MTBF（平均故障間隔）			30,000時間（ドライブ本体）
ビットエラーレート（訂正後）			1.0×10^{-12}以下
オーディオ出力レベル（インピーダンス）		ヘッドホン	0.18Vr.m.s（16Ω）
		ラインアウト	1.0Vr.m.s（47kΩ）
バッファ容量			256KB
使用環境	周囲温度	動作時	5℃～35℃
		非動作時	−20℃～50℃
	湿度（結露なきこと）	動作時	10%～80%
		非動作時	8%～90%
外形寸法（幅×高さ×奥行）			158mm×58mm×318mm
質量（本体）			2.5kg
SCSIコネクタ形状			ハーフピッチ50ピン
適応CD，CD-ROM			CD-DA，CD-G CD-ROM Mode-1, CD-ROM Mode-2 Form-1，Form-2, CD-ROM XA，PhotoCD，VideoCD, CD-IFMV

難問奇問の64ステージ

DRINKY&SMOKY PLUS

ドリームスタッフ

2月号で紹介した「DRINKY&SMOKY」が，仕掛けの増加，新マップ全64ステージ，背景グラフィックをパワーアップして「DRINKY&SMOKY PLUS」となって帰ってきた。

ゲームのルールは前作と同じ。基本的に直進するだけの主人公を導くため，ステージに仕掛けを置き，フィールドに落ちているすべてのウィスキーをドリンキー(またはスモーキー)に拾わせて,ドリンキーがゴールにたどり着けばステージクリアだ。

使える仕掛けは全部で10種類。

・いばら：風船を割るために使う
・標識：進行方向を変えるためのもの（3種類あって，1回接触すると消えるものと残り続けるもの，一方のみに進めるものがある）
・鉄骨：足場を作る
・風船：上方向に移動する
・扇風機：風を作る
・スモーキー：ドリンキーの弟
・消える鉄骨：キャラクターが上に乗ると一定時間後に消える
・ダイナマイト：キャラクターが通りすぎると一定時間後に爆発し，周り1キャラクタ分を吹き飛ばす(壊せないものもある)

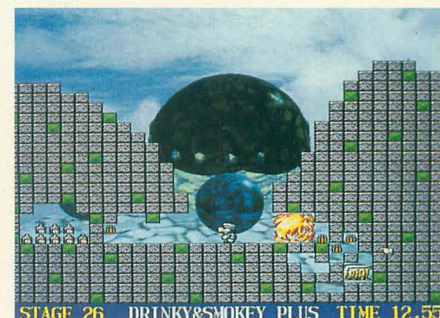
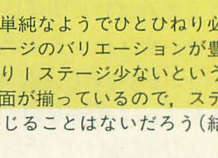
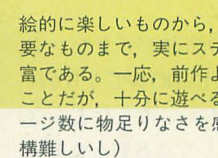
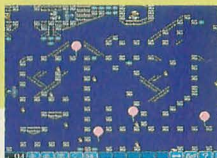
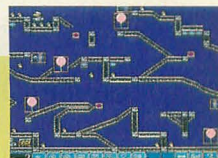
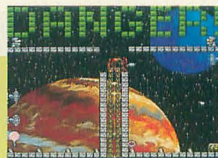
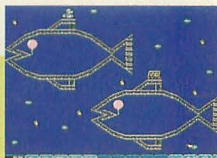
最後の2つが，PLUSから加わった仕掛けである。ちなみに，ダイナマイトの周りにさらにダイナマイトがあれば，当然誘爆する。この性質を利用した，発破好きにはたまらないステージも結構あるぞ。

さらにPLUSでは，ある程度ステージがセレクトできるようになった。セレクト画面にある8×8のブロック1つひとつがステージであり，1つのステージをクリアするとその両隣と下にあるステージが選択可能となるのだ。このシステムにより，詰まったときには別のステージからチャレンジできるようになり，徐々にマップを開いていく達成感が味わえる。

クリアしていくたびに新たな難問が……

そして，ステージごとに使用できるポイント，仕掛けは制限されているので，プレイヤーはかぎられた条件のもと，四苦八苦することになる。仕掛けも増え，マップが複雑化したため，ゲーム自体の難易度は，前作よりも上がっているだろう。クリアするまで1時間くらいディスプレイの前に座り続けることもしばしばあった。

とりあえず，クリアルートを見つけるだけでもひと苦労。次にクリアルートを見つけても，ポイントが足りなかったり，使いたい仕掛けがなかったりとひと筋縄ではいかない。

しかし，あの手この手を考え，試行錯誤しながら仕掛けをいじり，悩むのは結構楽しい作業だ。それに，難しいステージをクリアできたときの達成感も格別。実に遊び応えのあるゲームといえよう。

新しく加わったダイナマイトを有効に使え

〈購入方法〉

1,200円分(ゲーム本体1,000円＋送料200円)の無記名の定額小為替と返信用の宛先を書いたタックシール，そして希望するゲーム名を明記したものを同封して以下の宛先まで連絡すること。メディアは5インチ，3.5インチの両方をサポートしているので，メディアの種類も忘れずに明記しておくように。

〒260 千葉県千葉市中央区村田町280-2
若林俊夫方ドリームスタッフ

絵的に楽しいものから，単純なようでひとひねり必要なものまで，実にステージのバリエーションが豊富である。一応，前作より1ステージ少ないということだが，十分に遊べる面が揃っているので，ステージ数に物足りなさを感じることはないだろう(結構難しいし)

ム，ムズイ……

ゲーム全体の完成度は前作同様に非常に高いし，前作を踏まえて，うまくパワーアップさせている作りに好感がもてる。ステージ選択の「パワーモンガー」方式もいいね。

ソフトには，ヒント集がついてくるのだがこれは賛否両論。ま，あくまでも自分の力で解こうとするなら読まなければいいだけのこと。それにヒント集ということで，解答は書かれていないので，ちょっとくらい覗いてもゲームを楽しむのに支障はないはずだ。

結局，この「DRINKY&SMOKY PLUS」は，ステージ全体を通して，結構難易度が高い。そのため，さらに「DRINKY&SMOKY」を遊びたい，というような人には，文句なしにお勧めできる。で，興味はあるけどムズイのはかんべん，というような人は，まず前作の「DRINKY&SMOKY」を遊んでみよう。ゲームのルールを覚えつつ，サクサク進めるぞ(もちろん，最後のほうは難しくなるけど)。そして，もの足りなさを感じたら「～PLUS」にチャレンジすればいい。どちらにしても，じっくり考えるタイプのパズルゲーム好きの人なら絶対にハマるゲームだ。（浜崎正哉）

総合評価：★★★★★★★★★☆

多人数でわいわい遊べるクイズゲーム

クイズジョッキー

Midy House

近頃視聴者参加のクイズ番組というものがさっぱりなくなってしまったね。「アップダウン」「Q&Q」「タイムショック」「ドレミファドン」など。これらの番組が絶頂期には誰もがブラウン管にかじりつき「なぜ，こんなどこにでもいるようなオヤジが豪華賞品を！」などと羨み&ライバル意識を燃やしていたに違いありません。現在世に溢れるタレント解答者が雁首揃えたクイズ番組は，バブルの夢に破れてしまった人々へさらなる首のうなだれを強いるものにほかならないのです。自分が手に入れるチャンスのない賞品を，タレントが浮かれ気分でゲットするのを見てなにが楽しいのでしょう。これでは檻の外でバナナのむさぼりを見せびらかされているチンパンジーに等しいじゃないか！　というような理由ですっかりクイズ解答意欲を削がれてしまった若者には同人ソフトサークル「Midy House」制作の「クイズジョッキー」で友達ともども熱くなってみてください。

タイトルからわかるように，ゲームの舞台はいまはなき某テレビ番組が参考材料です。ゲームの進行がとことん番組仕立てに

関係ないけど□ート製薬のCMソングを口ずさんでいると「徹子の部屋」のテーマになっちゃうよね

基本的に早押しクイズがメインなので，問題を瞬時に読み取り，理解する速さが決め手となります

こだわって作られており，「タイムボカン系」終了後，即チャンネルを切り替えて「まんが日本昔話」から「8時だョ」までイッキに駆け抜けたサタデーナイト黄金パターン世代には大ウケ間違いなし。

プレイヤーは1～4人まで同時対戦可能。ゲーム自体には5人の解答者が参加し，足りない分は9人の個性豊かなコンピュータが担当してくれます。総問題数は1,698問あり，内容も変なマニアックさはないので誰にでも楽しめることでしょう。

問題解答形式はどうせ3択早押し一辺倒なんだろうですって!?　そいつぁ間違いってもんですぜセニョール！　早押し，連射早押し，特殊効果早押し，全員応答ペーパー，全員応答早押し，BET，ジャンル選択などなど多彩な解答形式がゲームの魅力を引き立ててくれます。

私はパソコンクイズゲームはこうあるべきものだと今回の遊技を通して痛感いたしました。正解すると女の人が服を脱いでしまうような「私脱いでもスゴいんです」系のクイズゲームなんてどうでもいいです(笑)。あと，対戦で避けなければならない事態が，答えを覚えてしまったX68000持ち主の圧勝ですね。こうなると持ち主はまるで「はらたいらさん状態」ですからね。

出演者と司会のボケ＆ツッコミもゲームを盛り上げてくれます。もう少しセリフのパターンがあるともっとよかったのに

〈購入方法〉

1,500円分の無記名定額小為替と住所，氏名，ほしいソフト名とメディア(5，3.5インチ)を明記したものと宛名シールを同封のうえ下記の連絡すること(このソフトはTAKERUでも販売されています)。

〒604　京都府京都市中京区壬生梛の宮2-2
小原方 Midy House

やっぱり多人数で遊びたい

本文中で述べているように，一貫してテレビ番組を意識した作りがいいです。オープニングに始まり，出演者紹介，番組提供テロップにCMまであり，とても凝っています。

多人数でわいわい遊ぶのに適していますし，ひとりで遊んでもちゃんと楽しめます。

また，優勝するとスペシャルクイズにも挑戦できるようです。このへんの付加価値もいいですね。本当にクイズ番組が好きな人にはお勧め。

ひとつだけ問題があるとすれば，出題された問題と解答の選択肢が上下に分かれていて非常に見づらいという点でしょうか。いや，見づらいというより，問題と選択肢の間で目が動いてしまい，ついついボタンを押すのが遅れてしまうのです。ここは，問題を読み上げるがごとく1文字ずつゆっくり表示し，問題を表示し終わったら選択肢を表示してもらったほうがよかったかな？　でも，いかに速く相手よりも問題を理解しなければならないというプレッシャーがあって，これはこれでよかったのかもしれませんけれどね。

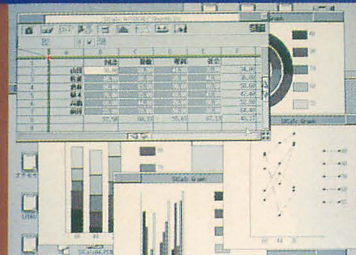
(須藤芳政)

総合評価：★★★★★★★★☆☆

THE USER'S WORKS SPECIAL

SX-WINDOW用表計算ツール

SX CALC

北谷 宇一

SX CALCの基本的な使い方は，巷にある表計算ツールだいたい同じ。セルと呼ばれる縦横に区切られたエリアに数値(または関数など)を書き込んでいき，目的の結果を得られるようにすればいい。

このSX CALCでは，セルをX方向，Y方向ともに32,767個まで扱える(といっても，これは理論上の話。現実にはメモリ容量に制限される)。そして，このセルのなかには定数，文字列，式，ほかのセルの内容，関数などが入力可能だ。

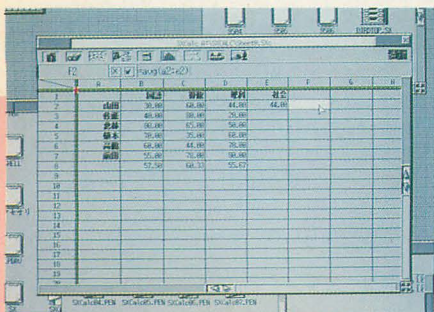
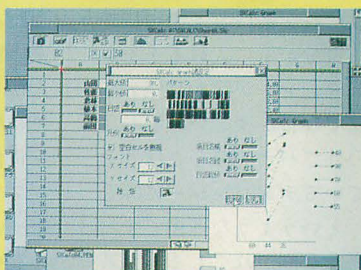
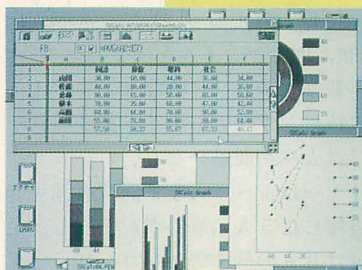
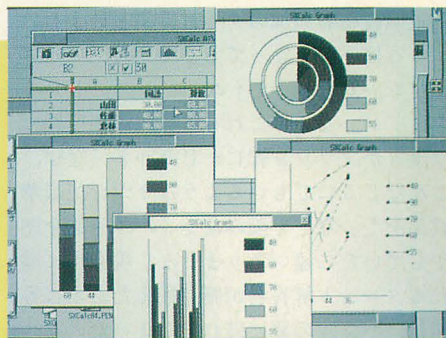
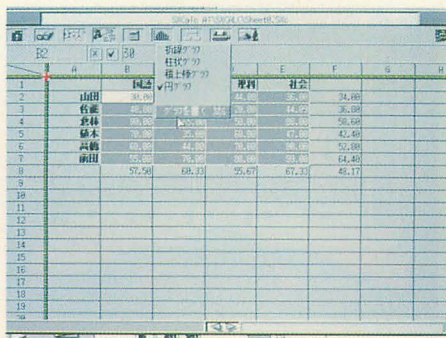
たとえば，A1～A4セルの平均をA5セルに算出させるとしよう。この場合は，まずA1～A4セルに数値をサクサク打ち込み，そしてA5セルに，

=AVG(A1:A4)

以上のように書き込めばいい(合計を求めたければSUM()関数を使う)。

また，セルは幅や高さの変更，セルを並べ変え(ソート)，挿入，削除，フォントの種類や大きさをセルごとに変える，表示する書式や桁揃えなどを設定できる。過不足なく，だいたいのことに対応しているのが嬉しい。

入出力ファイル形式としては，SX CALC独自の情報を盛り込んだオリジナル形式，SYLK形式，CSV形式(各要素を"で囲み，,(コンマ)で区分けしたもの)や，タブテキスト形式(各要素をタブで区分けしたもの)をサポートしている。ただし，SYLK形式については，BUSINESS PRO-68Kで使われているデータを独自に解析してあるため，完全に互換性があるかどうかは疑問(と制作者自身がいっている)。

そして，専用のマクロシートを用いて簡単なコマンドマクロも実行できる。ただし，マクロはアイドルイベントが発行されたときのみ行われるので，ちょっと実行速度が遅い。

とりあえず，グラフは4種類。これからいろいろ増やしていくということだが，ぜひともグラフの任意サイズ描画もサポートしてほしい

グラフは，現在折れ線グラフと柱状グラフ(ヒストグラム)，積み上げ棒グラフ，同心円グラフの4種類をサポート。これらはPICT形式でコピーできるので，シャーペンやEasydraw SX-68Kなどに張り込める。ただ，グラフ描画部分が小さいので，複数の項目を一度に表示しようとするとかなり見づらい。

印刷関係では，印刷結果のプレビュア機能，そして結果をPICT形式でコピーすることができる。これも嬉しい機能。

以上，すべてではないが，SX CALCを簡単に紹介してみた。なお推奨環境はSX-WINDOW ver.3.0以上＋X68030＋4Mバイトメモリ(コプロもあればなお可)である。

〈入手方法〉

このプログラムは，NIFTY-Serveのシャープフォーラムにアップされている。なお，シェアウェアとなっているので，試用してみて気に入ったのであれば，ドキュメントに書かれた方法に従って送金すること(シェアウェア代金6,800円)。

う～ん，見るからに表計算(そのままやんけ)

バージョンアップに期待

SX CALCは，オンラインマニュアルもしっかりしているので，特にスプレッドシードに関する知識がなくとも最低限の機能は理解できる。操作性についても変なクセはなく，実に素直な作りだ。突出して出来がいいところはないが，致命的な問題もない。平均的によくできている。

そして，SX CALCのいちばんの欠点かつ利点は，完成品ではないということ。これはマニュアルを見ればわかるのだが，フォーマットの変更が予告されていたり，不安定な動作を黙認していたりする(いまのところ妥協しているところが多い)。つまり，まだ発展途上の段階であって，まだまだやらなければいけないことを抱えているのだ。これは，逆にいうと完成するまでに問題が解決される可能性が非常に高い。というより，実際に解決しようとする意志が制作者にはあるので，問題とはならないのかもしれない。

結局，SX CALCの制作に参加するというくらいの心構えで購入し，制作者と一緒になってよりよいものを作っていただきたい。

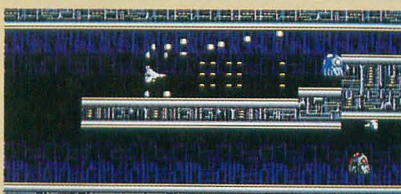
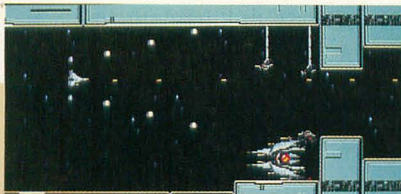
総合評価(現段階)：★★★★★★☆☆☆☆
総合評価(完成予想)：★★★★★★★★☆☆

弾よけが熱い！

Griffon

BI-Factory

BI-Factoryの「Griffon」はX1turboZ以降専用のアイテムパワーアップ型の横スクロールシューティングだ。基本的にグラディウスにイメージが近い。全7ステージは砂漠から宇宙までレパートリーに富んでいてなかなか楽しめるものに仕上がっている。

さっそく起動してみるとまずは美しいタイトルのフェードイン&アウト。うーん，この辺はさすがアナログパレットだよなあ…(しみじみ)。さっそくゲームを始めてみよう。武器は敵の落とす武器チェンジアイテムでManyWay, Laser, Homing, Illusionの4つから選べるようになっている。もちろんすかさずLaserを選択する（どうも私はレーザーオタクの気がある…)。各武器はこれまた敵が落としてくれるパワーアップアイテムで，3段階にわけてパワーアップしてくれる。またR-TYPEのようなため撃ちもできるようになっていてなかなか奥が深い。画面奥から現れる敵もいたりして演出も凝っているといえよう。そして面の最後にはお約束のボスキャラ。なかなかビッグサイズなボスキャラで，プログラマの力量を感じさせてくれる。では簡単に各面を紹介していこう。

1面　Night Desert

簡単に肩慣らしといったところ。でも中ボスがいたりして結構やっかい。

2面　Space Colony

いわゆるひとつの通路面。後半でのレーザー&弾よけはかなり熱い。

3面　Asteroid

隕石面。この面から地雷やら，曲がるレーザー（私はこれが苦手）で攻撃が多彩になってくる。

4面　Rock Cave

通路の狭さに加えての激しい攻撃でかなりの難面。中ボスの岩石親分&子分（勝手に命名）に何度やられたか……。

5面　Desolate Shrine

ソーサリアンのような宮殿面。この面の中，大ボスはすごい！　特に大ボスはまるでスプライトのようだ。

各面にはそれぞれ中，大ボスが配置されている。攻撃は多彩かつ熾烈を極め，プレイヤーのシューティング魂を熱くしてくれるだろう

6面　Space Dancin'

宇宙面。これも途中に出てくる中ボスの動きが美しい（攻撃は鬼のようだが)。

7面　Mega City

最終面に登場するボスはなんと画面の半分近い大きさ。うーん，やるねぇ。

さて今回編集部に届いたディスクは85%完成バージョンということで，まだちょっとバグがあったり，BGMがどっかで聴いたことがあるものだったりするが，全体的に完成度は高い。たった2人でここまで作れるのはすごい，というのが素直な感想だ。ぜひ完成版をPLAYしてみたいと思わせる入魂の1作。開発ぜひがんばってね！

まだまだ序の口。背景を楽しみながら進もう

基本的に武器の切り替えはアイテムを使って行う。ただ，今回デバッグモードとしてついていた，武器の切り替えボタンがかなり使いやすかった。ゲームの難易度が高い分，武器を自由に切り替えることによってプレイヤー側の負担を軽くすることもできるので，このままシステムに残してくれると非常に嬉しいのだが(難易度によって使えるようにするとか)

完成が楽しみ

X1turboZ以降ならではの総天然色4096色（懐かしいフレーズ！）で描き込まれた背景，キャラクターはさすがに美しい。背景も滑らかにスクロールしているので見ていて実に気分がいい。敵の性格づけなんかもよく考えてあって，攻略法を考えながら進む楽しみもある。特に5面の中，大ボスは個人的にお気に入りだ（しかし曲がるレーザーには本当に泣かされた)。

ただし，めちゃくちゃ難しいのである。とにかくスピード。もうほんとに速い！（ザコと中ボスが入り乱れるとさすがに重くなったりするが)。あのスピードで弾よけを強いられるのはちょっと辛いかもしれない。ただし，当たり判定がBOXではないので，SUPER LAYDOCKな弾よけができるのは熱い！

全体としては，弾よけの大好きなハードシューターには，たまらないゲームといえる。とはいえ，軟弱な私としては，もう少しランクを落としたモードもつけてほしいな。　（高橋哲史）

パズル要素を加えたRPG

PUZZ MAZE

MoSstation

このゲームは，主人公が何者かによって導かれたモンスターの溢れかえる遺跡を探索しながら帰り道を探す，というストーリーをもつ3DダンジョンタイプのRPGである。ゲームの舞台は，遺跡とその近辺にある町だ。町にはRPG定番の病院，鍛冶屋，道場，アイテム売り場，そして酒場兼宿屋がある。それぞれの場所にいるキャラクターは，なんだか雑多でまとまりがなくこれといった特徴のない者ばかりだが，酒場にいるカマっぽいマスターは，ちょっとだけ気に入った。

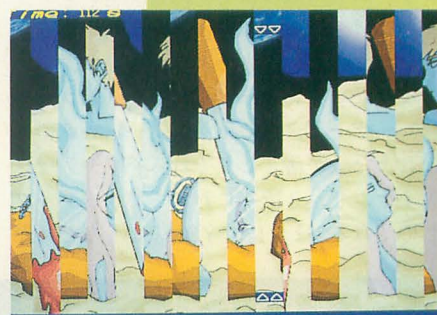
モンスターとの戦闘はごく普通のターン制によるもので，コマンドも剣による攻撃，魔法攻撃(1種類)，各種アイテムを使う，といったシンプルなもの。レベルアップシステムは，いわゆる経験値によるものではなく，モンスターを倒したときにもらえる金を貯めて，遺跡と町を行き来しながらセコセコ鍛錬を積むようになっている。道場に行けば体が鍛えられ，鍛冶屋に行くと剣を鍛えてくれる。

ここで，注意してもらいたいのは，ついつい道場に通いつめ，体力バカになりがちだということ。基礎体力も大切だが，剣を鍛えておかないと攻撃の命中率は上がらないことをよく覚えておこう。

そして，遺跡の各所には扉があり，そこには謎のボスにマインドコントロールされた人間(なぜか女の子ばかり)か，ガードシステムが存在している。マインドコントロールされた人間は，いわゆる中ボスみたいな存在であって，どこのRPGでもよく見られるものだ。で，面白いのがガードシステム。いわゆる侵入者防止システムという設定だが，扉のロックを解除するためには，決められたパズル(全部で3種類)を解かなくてはならないのだ。

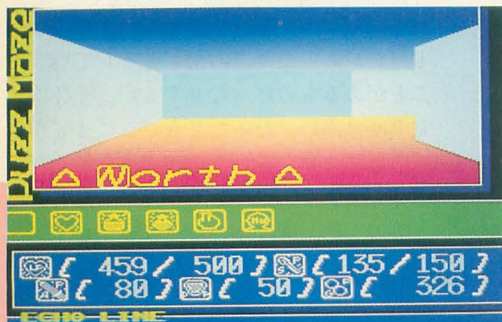
このへんの要素を強調したくてタイトルに反映させているのがよくわかる。ところが，別にパズルを解かなくても扉にあるガードシステムと戦闘をすることができ，勝てば強引に扉を蹴り破って進むこともできたりする。ただし，結構手ごわいので最初のうちは嫌でもパズルを解かなくてはならないのだが，レベルを上げていくと余裕のよっちゃんで倒せてしまうのだ。

結局，パズルという要素が生かされておらず，ただの思いつきで終わっているのが非常に残念だ。

ヒントを頼りに該当するXを探すSCROLL

パーツを入れ替えて元絵を再現するPARTS

〈購入方法〉

1,000円分の無記名定額小為替と住所，氏名，ほしいソフト名とメディア(5, 3.5インチをサポート)を明記したものと宛名シール(なくても可)を同封のうえ，下記の住所まで連絡すること。

〒791　愛媛県松山市山越6-10-10　坂本和秀方 A.S.G.

遠くまで見渡せそうで見えない3Dダンジョン。ただし，建物の上部が吹き飛んでいるという設定らしく，空が丸見えで気持ちいいかもしれない。あと，戦闘シーンと町の病院のお姉さん。お近づきになりたかったのだが，ケガは魔法で治したほうが安上がりなのでほとんど利用せず(涙)

う〜む，もうひと息

全体的にゲームデザインが雑である印象を受ける。雑というよりゲーム作りに慣れていない，もしくは自分本位でゲームデザインをしてしまっている部分が目立つ。理解に苦しむようなメニューが表示されたり，イベントが発生してもなにがなんだかわからないところがあり，素直にゲームを楽しめない面がある。

また，遺跡のマップもそれほど凝っていないし，トラップといったものも存在していないので，実に単調なゲーム展開になりがちなのも問題だ（これは，戦闘がほとんどスペースキーを押しっぱなしでOKなのも原因だろう)。お楽しみのパズルもあってなきがごとし。

ただ，未熟ながら一応完結しているところは評価できる。X68000上で初めて作ったゲームとしては，よくまとめ上げたといいたい。あとは，ゲームを客観的にプレイしてくれる友人を見つけ，これからも，もっともっとがんばってもらいたい。（浜崎正哉）

総合評価：★★★☆☆☆☆☆☆☆

THE USER'S WORKS大募集!

今回，THE USER'S WORKS SPECIALということで，同人サークル，個人制作のソフト５本を紹介しました。

パズル，クイズ，シューティング，RPGに表計算ツールとバラエティにとんだラインナップです。

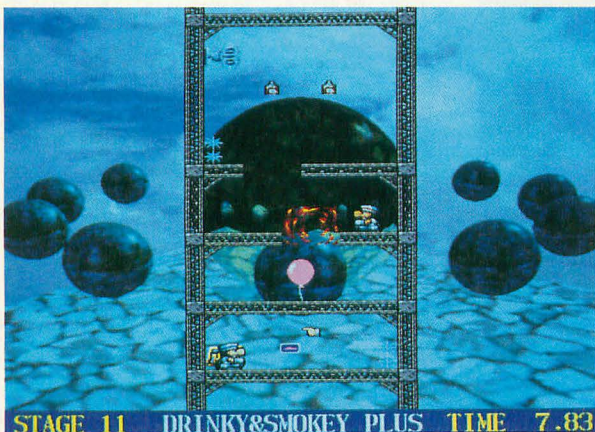
ただし，ソフトのレベルについてはかなりの開きがあります。多人数で制作しているところは比較的作品の質は安定しているといえますが，やはり同人ならではの甘さを抱えているものも少なくありません(なかにはプロにいた人たちが，仕事の合間に作ったものもありますが)。

読者の皆さん自身が，記事内容，評価点数を参考にして，購入するかどうかじっくり考えてみてください。扱われている記事の大きさも評価のポイントとするといいでしょう。そのうえで，納得できるものであろうと判断したら，さっそく連絡を取ってみてください。

また，評価については，結構厳しく設定しました(市販ソフトよりはやわらかいけど)。いかに値段が安くても，あくまでもお金を取って販売する以上，ただソフト情報だけを掲載するわけにはいきませんからね。

そして，購入の注意点としては，住所，氏名は当然のことながら，ほしいソフト名，メディアを明記すること。そして送金方法は，必ずサークルから指定されたとおりに行ってください。

あと，あまりにもソフトが届くのが遅すぎるなどの理由で連絡を取りたい場合は，必ず返送手段を用意したうえで行うようにしましょう(往復ハガキで問い合わせる，返信用切手を同封するとか)。あくまでも個人どうしの取り引きですから，信頼関係が大切です。

そして，THE USER'S WORKSでは，これからもいままで以上に広く定期的に作品を紹介し続けたいと思っています。X68000における同人ユーザーのパワーを見せつけるチャンスでもあります。完成度によっては，完成品でなくても情報を掲載することもあります(Griffonがそうですね)。また，作品以外にもなにかご意見がありましたら，遠慮なくアンケートハガキでお寄せください。それでは，皆さんの力作をお待ちしています。

SOFTWARE INFORMATION

今月は新作情報なし，ということでちょっとTHE USER'S WORKSを間借りして，ソフト状況をお伝えしていきましょう。

まずは，電波新聞社の「バラデューク」。こちらは予定どおり発売されました。すでにグログロなモンスターたちとたわむれている人も多いことだと思います。

アンソロジーシリーズも順調に発売され，気になる次回作は……いったいどうなるのでしょうか。

続いてリーズナブルなお値段，かわいいキャラクターが魅力のアドベンチャーシリーズ「EXCITINGみるく」です。こちらの制作はまずまず順調のようです。来月号では画面写真を公開できる……かな。

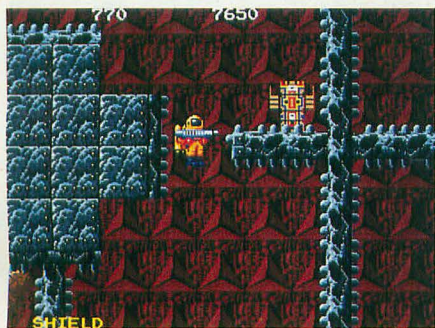
また，TAKERUではかなりの同人ソフトも取り扱っています。興味のある方は一度覗いてみるといいでしょう。思わぬ掘り出しものがあるかもしれませんよ。

過去の作品を安価に提供する名作文庫シリーズもTAKERUの魅力のひとつ。今回新作リストはありませんが，今後どんなソフトが収録されるのかちょっと楽しみですね。

「地球防衛MIRACLE FORCE」「プリンセスメーカー」は，とりあえず夏以降を楽しみに待て！　という状況です。そのほかのソフトについては動きが見られず。う～む，さびしい。

発売中のソフト

ソフト名	メーカー	発売日
★バラデューク	電波新聞社	5/26
X68000用	5″2HD版	5,300円(税別)

新作情報

ソフト名	メーカー	発売日
★EXCITINGみるく	TAKERU	7/未
X68000用	5″/3.5″2HD版	1,500円(税込)
★X CASE	Beシステム	
X68000用	5″2HD版	19,800円(税込)
★Traum	象スタジオ	
X68000用	5″2HD版	価格未定
★麻雀悟空・天竺への道	シャノアール	
X68000用	5″2HD版	9,800円(税別)
★地球防衛MIRACLE FORCE	カスタム	
X68000用	5″2HD版	価格未定
★プリンセスメーカー	ニュー	
X68000用	5″2HD版	14,800円(税別)

祝！ バラデューク10周年

ヌルヌル，ドロドロの怪生物たちを蹴散らしながら突き進む「バラデューク」。シンプルながらも，さまざまなフィーチャーが，ゲームを盛り上げてくれる。かわいいパケットと協力してプレイするか，銃殺するかはあなたしだい。

Yaegaki Nachi
八重垣 那智

失礼な話かもしれないが，最近よく見かけるリバイバル風の続編ゲームについて疑問に思うことが多々ある。つまり，現在のプレイヤーが，そのオリジナルをどれだけ知っているのか？　ということである。7年から10年以上も昔のゲームから名前だけもってきたような，似ても似つかないゲームに腹を立てていると，目の前の偽物がオリジナルのファンどころか，制作者ですらないがしろにしているようにしか見えなくなってしまい，悲しい気分になってしまうのだ。ファミコンの移植版ですら10年も前に出たような，安易な名前だけの続編の制作者には，オリジナルの貴さと，そのゲームの面白さを100%理解したうえで，制作に打ち込んでもらいたいものである。

恐ろしさに慄く

今月の素材は定期的にX68000ゲーム市場に油をさし続けてくれている電波新聞社のアンソロジーシリーズの第13弾「バラデューク」である。なにも足さない代わりになにも引かないのが，このアンソロジーシリーズのモットーなのだそうだから，冒頭に書いたような不満の心配の必要もなく，昔のままのゲームが遊べる点については，安心してよいだろう。

まずは簡単にこのゲームの履歴から紹介していこう。メーカーは当時最盛期を少々過ぎていたナムコで，デビューは1985年の夏。それまで類を見なかった，リアルタッチのモンスターによるヌルヌル感が画面上に表現された，ドロドロなゲームであり，このゲームに対して上記のような記憶をもつ人は少なくないだろう。もちろんその記憶は，おおむね間違いないといっていい。いまでこそ質感表現の手法と画像ハードウェア技術の発達により，これが陳腐に見えることがあろうとも，当時の非常に鋭いインパクトが，このゲームを見る人の心に消えることのない深い記憶を刻みこんだことがわかる。

パケットはできるだけ助けよう！

ゲームのルールは，各フロアに点在するオクティと呼ばれる青い怪生物を倒すことである。この怪生物に支配された星を救う任務のため，黄色の宇宙服のようなものを着たプレイヤーを操作していく。レバーで8方向に移動し，ボタンで向かっている左右どちらかに攻撃を行う。画面はプレイヤーの移動によって，フロアごとに異なる構造をもつ内部を任意にスクロールするようになっている。ただ，基本的には各所に配置されたオクティを全滅させ，ステージ最下部の脱出口から脱出すれば，そのフロアはクリアとなる。

ステージはそういった通常面4つと，息抜き的要素の強いオクティのいないアイテム回収面，それに巨大なグレートオクティと対決するボス面の，計6フロアで構成されている。ゲーム全体は計8ステージ48フロアによって構成されており，さまざまなフロアとオクティが牙をむいてプレイヤーを待ち構えている。プレイヤーはこれらに，銃のみで戦いを挑んでいくが，不幸にして敵の弾や敵本体と接触するとシールドを失い，手持ちのシールドがなくなると1ミスになる。つまり手持ちシールドが2つの場合，2発食らうと1ミスになる。

ここで特筆すべきことは，プレイヤーは常に一定の力で画面下方向に弱い重力の力で引かれていることと，銃を発射したときに逆の方向，つまり左右に反動で戻されるという2つの点である。この2点がバラデュークにおいて，ゲームにおける独特の存在感を生み出しており，さらにこういった特徴をゲーム進行中に，プレイヤーへ意識させるといった細かい演出がなされている。

X68000用　5"2HD版　5,300円(税別)
電波新聞社　☎03(3445)6111

ムネン，アトヲタノム……すまないねえ

謎の顔が出現！　迷わず倒そう

埋まっているのではなく，実は抜け道

日本で一番狭い面。西日良好風呂便所なし

ここで待ってもフルーツは出ない!?

緻密さに酔う

バラデュークでは，敵であるターゲットのオクティは固定配置されている。各ステージは基本的に迷路になっているので，それらを効率よく退治するルートを把握することが，まずは重要になる。中盤以降は，銃の反動で壁を抜けたり，隠し通路の活用や複数の出口の選択といった部分も，知識的攻略要素に入ってくるので，これらは繰り返しプレイして調査したり，知識を身につけなくてはならない。地形と敵のマップ配置の妙が味わえる，当時のナムコゲームの特徴は，こういった部分に顕著に表れているようだ。

ところが，オクティ以外の敵はほとんどランダムに発生したり出現するようになっていて，知識と訓練による正確なプレイからは離れた印象を受ける。雑魚敵はステージによってその種類が限定されてはいるのだが，あくまでも出現はランダムである。ここでプレイヤーは，オクティの護衛的な役割をもっているこれらの雑魚敵を，いかに処理するかという臨機応変の対応が要求される。結果，マップを覚えた程度で済むようなパターンゲームにならないように，工夫されているといえるだろう。

また雑魚敵だけでなく，オクティを倒したあとに出現するカプセルの中身もランダムである。仲間となる黄色の生物であるパケットが現れればシールド増加チャンスになり，銃のパワーアップといった珍しいアイテムも隠れているが，中から敵が出現しダメージの危機にさらされることもある。

これらのランダム的な要素は，上級者，初心者に関係ない部分なので，運がよければ初心者でもフルパワー，フルシールドになることができ，思わぬ先の面まで行けたりする。これは，意識的にあえて入れてあるもののようだ。その半面，繰り返し挑戦するプレイヤーには練習の成果が素直に反映されないようなことも起こりうる。そういった意味ではゲームに対してシッカリした手応えがないという，不満を感じるプレイヤーのいるようだ。当時，目立ってはいたもののあまりヒットしなかったのは，これらの部分とあながち無関係ではないだろう。しかし，確実さのためにランダム的な展開も想定した攻略を考えると，さらにそれ以上の緻密なプレイが要求されるという点が，実はこのゲームの奥の深さではないかと思うのである。

厳しさに痺れる

ついゲームそのものの説明が長引いてしまったが，X68000版の具合も忘れずに見ておくことにしよう。今回もいつもどおり，売り文句は完全移植である。本来横画面のゲームであるため画面周りの変更は一切なく，画面の印象は寸分違わないものである。各種フィーチャーなどもキッチリ移植されているようだ。特定のフロアにおける隠れキャラクターや，ネーム入れのレインボー処理といったものまで，きちんと押さえてある。見た目については問題ないといってもいいだろう。

しかし，このゲームで最も重要なのはそういった見て調べられる部分ではない。長々と書いてきたように，このゲームを支配しているのは，表面に出てこないランダムに操られた部分であり，そういった部分のフィーリングまでも同じだと断言することはできない。ただ，バラデュークをしのび，またこのゲームを知らなかったプレイヤーが，そのなんたるかを知りうるには，少しも問題のない出来である。アンソロジーシリーズが，旧作の復権をお題目に掲げている以上，これは立派にその責を果たしていると考えるべきだろう。

この移植に対し，オリジナルの基板と比べて画面外の処理が違う，敵の発生タイミングが違う，などというような指摘をするプレイヤーは確かにいるかもしれない。しかしそういった違いを追及するならば，それこそオリジナルの基板を入手するべきであり，X68000に移植されたもので満足するような人種ではないことを自覚するべきだ。オリジナルあっての移植であるわけだから，本物至上主義であるのはもっともなことだ。しかし，移植は移植なりの存在意義があるのである。

懐かしさに和む

そう納得してプレイすると，これはこれで楽しめるものではある。とはいってもプレイヤーの攻撃が地味で自由度が高すぎるせいか，やはり最近のゲームと比べると難しいという印象を受ける。そういった部分も含めて，このバラデュークを10年目のいまになって再評価する価値は十分にあるだろう。1985年という年は，風俗営業法の施行に伴いゲームセンターというものに転機が訪れた年でもある。そんな歴史も考えながらプレイしたいものである。

深く静かに戦闘せよ

このゲームは見かけによらず，細かいテクニックの宝庫だったりするのですが，そういったものの示唆が画面から一切ないあたりに，時代を感じることができます。当時はなんの疑念もなく純粋にゲームを楽しめたのですが，さすがに10年経ってから振り返ると，気に入らない部分が多々あるのはしようがないのかもしれません。当時並行してやっていた，出たなポップのパラダイスでヤッホーなシューティングに比べると，こちらは流行に取り残された感じがしますが，時代を作った存在として，昔のままで毅然として残ることは，逆に貴いことなのだと思います。いつになく真面目だな，うんうん。

総評	0 5 10
ゲーム性	★★★★★★★
グラフィック	★★★★★★★
技術	★★★★★★★
サウンド	★★★★★
ドロドロ	★★★★★★★★
パケット銃殺	★★★★★★★★★

下は海・・・

DōGA CGアニメーション講座 ver.2.50……………………………(第25回)

変形グニャグニャ(その2)

プロジェクトチームDōGA
かまた ゆたか

前回に続いて物体変形ツールの紹介です。今回はBOXTRANS.X を使ってみます。これは意外と使えるツールのようですので,「Graphic Gallery」のような面白い画像を作ってみましょう。

はじめに

PROJECT TEAM DōGAも長年活動しているせいか,平均年齢がだんだんアップしてきました。これは先細りの前兆ということで,この4月から,大阪大学コンピュータクラブの1年生への指導,教育を強化しています。いろいろな人が参加してくれるのはよいのですが,なかには相当な初心者もおり,面白い発想をしてくれます。

たとえば「HDD,ディスプレイ,本体の順に電源を入れて」というと,キーボードをじっと見つめ「これの電源ボタンはどれですか?」と聞くし,「受け取ったディスクにはラベルを貼って」と渡せば,表と裏の両面に貼ったりします。

確かに,我々にとっては当然のことでも,初めての人にはわからない問題がたくさんあります。CGAがもっと多くの人に広がるためには,そういった「自分勝手な当然」を排除しなければいけないのかもしれません。

さて,前回は実用性の少ないツールEXPOINT.X を紹介してひんしゅくを買いましたが,今回のは実用性が結構あります。お世辞にも使いやすいとはいえませんが,がんばって使ってみましょう。

図1 元形状

図2 直方体領域の設定

図3 直方体の頂点移動

図4 物体の変形

BOXTRANS.Xの基礎

CGAシステムの物体変形ツール第2弾のBOXTRANS.Xは,EXPOINT.Xが特定の1つの頂点を移動させるのに対して,直方体領域に含まれるポリゴンすべてを変形させるツールです。

図が難しくなるので2次元で解説しますが,図1のような形状があったとします。それに対して図2のような直方体領域を設定します。そして,その直方体の各頂点を図3のように移動させます。すると,図4のように直方体領域の中が変形します。

直方体領域の設定とその頂点の移動量は,コマンドファイルで指定します。例によって,このコマンドファイルをエディタで記述しないといけないのですが,EXPOINT.Xよりずっと簡単ですので,先に解説してしまいましょう。

リスト1を見てください。まず「box」は直方体領域を指定するコマンドです。直方体領域のどれか1組の対角に位置する頂点の座標を与えます。パラメータが6個ありますが,前の3つと後ろの3つがそれぞれの頂点のX,Y,Zの座標です。わかりにくければ,

box(Xの最大値 Yの最大値 Zの最大値 Xの最小値 Yの最小値 Zの最小値)

と考えても問題ありません。

次に「vector」は,直方体領域の各頂点がどれだけ移動するかを指定しています。直方体の頂点は8つあるので,各頂点のX,Y,Z方向の移動量ということで,3×8=24のパラメータがあります。

ただ,ここでひとつ問題があります。何番目のパラメータが,どの頂点の移動量を指定しているのか,さっぱりわからないという問題です。そこで「vector」では,図5のように指定する頂点の順番を特定しています。つまり「vector」の最初の3つのパラメータは,必ず直方体領域のX,Y,Zの各軸方向とも+側の頂点の移動量を示し,次の3つは,Xが−で,YとZが+側の頂点を意味しているわけです。以下同様です。

ちょっとわかりにくいし,マヌケな仕様ですが,とり

あえず実用性はあります。

ついでに解説すると，「vector」の代わりに「boxtrans」というコマンドを使用することができます。使い方はほとんど同じで，24のパラメータをずらっと並べます。違いは「vector」が各頂点の移動量を記述するのに対して，「boxtrans」は移動後の具体的な座標を指定するだけです。どちらを使っても，結果はまったく変わらず，手間もほとんど同じでしょう。

以上がコマンドファイルのすべてです。これ以上のことはなにも記述できません。つまり，同時に複数の領域に対して変形を加えることもできません。ということで，EXPOINT.Xのように悩むような点はありませんので，さっさと実用編に移りましょう。

モデリングへの利用

BOXTRANS.Xの使い道としては，まずモデリングがあります。これは，意外と実用性が高いので，修得しておいて損はありません。

CAD.Xを使ってモデリングをしたことがある方は経験があると思いますが，X，Y，Zの各軸に平行な面だけから構成された物体は非常にモデリングしやすいのです。しかし，それぞれが微妙に斜めになっている面の物体は，各頂点が中途半端な値になり，最近点やら平面投射の機能をなんども使った煩雑な作業になってしまいます。そして，苦労して作っても，作っている間に，だんだんバランスが崩れてきて，なんかへんちくりんなデザインになることもよくあります。

そういうときこそ，まずX，Y，Zの各軸に垂直な単純な形状を作り，それをBOXTRANS.Xで歪ませることで，微妙に傾いた形状を作るという使い方があります。そしてもうひとつは，最後になって，微妙にバランスが崩れていることに気がついたときに，そのバランス調整にBOXTRANS.Xを使うという方法もあります。

たとえば，写真1のような形状を作ります。この形状は，すべてがどれかの軸に垂直なので，簡単にモデリングできます。これを，先ほどのリスト1のコマンドファイルで，変形させます。このコマンドファイルは，図6のように，前2/3を引き延ばし，すぼめるという変形です。

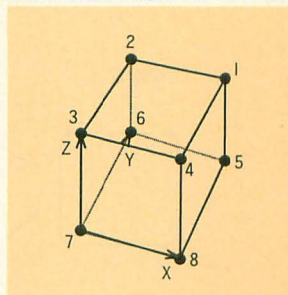
その結果が，写真2です。ご覧のような形状を最初からモデリングするより，BOXTRANS.Xを使用したほうがはるかに簡単です。そして，変形後のバランスが悪ければ，コマンドファイルのパラメータを変更して簡単にやり直せますが，CAD.Xでモデリングしたなら，手直しのしようがありません。

しかし，問題点が2つあります。ひとつはこの写真からはわからないのですが，BOXTRANS.X実行後の形状は，すべてのポリゴンが三角形に分割されてしまいます。

リスト1　コマンドファイルの例

```
box( 250 200 250  -50  -200 -220 )
vector(
        150      -80      -50
        0        0        0
        0        0        0
        150      80       -50

        130      -100     150
        0        0        0
        0        0        0
        130      100      150
```

図5　頂点の順番

変形後が明らかにひとつの四角形になる場合でも分割します。これは，ポリゴン数が増えるなど喜ばしくないのですが，現在のところ回避する手段はありません。まぁ，写真を見てのとおり，レンダリングの結果にはほとんど影響を与えません。

もうひとつの問題は，写真2の図形がよく見ると図6と異なっているという点です。図6では，直方体領域内は斜めになっているものの，それ以外の部分はなんら変形していないのに対して，写真2では，側面のすべてが斜めになっています。

これを，図7で解説すると，頂点Aは直方体領域になるので移動の対象になるが，頂点Bは領域外になるので移動しません。頂点AとBの間には頂点がありませんので，できる面はABを結んで，直方体領域外でも斜めになるわけです。

この問題を解決するためには，AB間の直方体領域の境の位置に頂点を1つ置いてやればいいのです。しかし，

写真1　基本形状

写真2　図7のような変形

図6　リスト1の変形

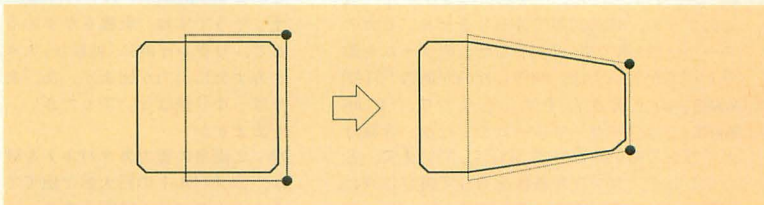



図7　領域外も変形する理由

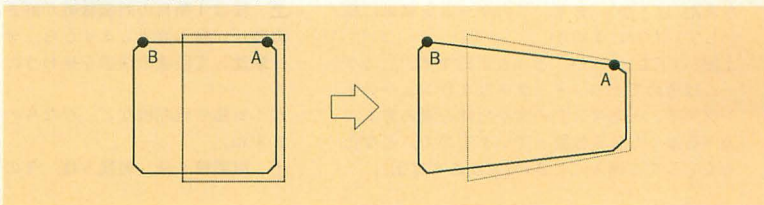
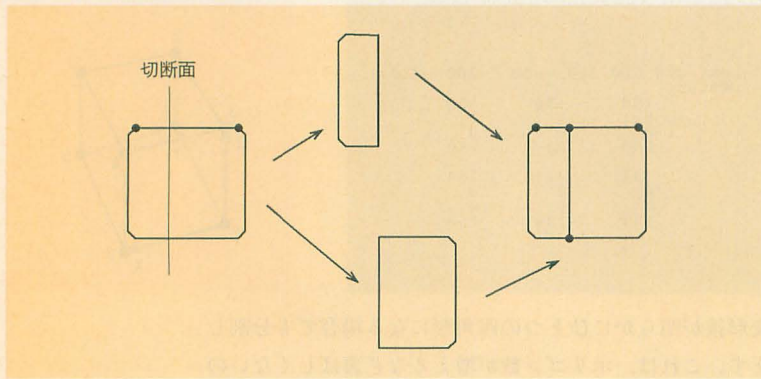



図8　ZANTE.X を使う

まじめにCAD.Xで修正していくのはあまり現実的ではありません。そこで，ZANTE.Xを使ってみましょう。

ZANTE.Xとは，任意の平面で形状を切って，別々の形状にして出力するツールです。第6回CGAコンテストのオープニングで，宇宙戦艦がビーム光線によってまっぷたつになるシーンなどにも利用されています。このツール，どうやって切断面を指定するかといえば，その形状の中の「atr no」のポリゴンで指定します(ご存じのように，CGA共通規格では「no」という名前は一般のアトリビュート名に使えません)。

たとえば，「TEST.SUF」という形状があり，それを2つに切る場合，まず「TEST.SUF」をCAD.Xで読み込み，アトリビュート名を「no」にしたまま，切断面に相当するポリゴンを1つ作ります。このとき，このポリゴンは，物体より大きかろうと小さかろうと，四角形だろうと三角形だろうと構いません(頂点が3つあれば1つの無限大平面が確定するからです)。そして，セーブして終了します。そこで，

ZANTE TEST /OCUT

とでも実行すると，「CUT001.SUF」と「CUT002.SUF」が出力されます。この2つのファイルは「TEST.SUF」を切断したものです。

同様に，図8のような位置に切断面を置き，ZANTE.Xで2つの形状に分断し，それをCAD.Xで再びアペンドします。すると，直方体領域の境に全部頂点が生成されるわけです。

ちょっと手間がかかりますが，1ポリゴンずつ修正することを考えたらとってもお手軽な手法です。このような処理を加えたあと，リスト1の変形を行った結果が，写真3です。ご覧のように，直方体領域以外の部分はまったく変形が行われていません。

ついでに写真4は，一度BOXTRANS.Xで変形させた物体について，さらに後ろ1/3を別のコマンドファイル

写真3　ZANTE.Xを使い図6の変形を実現

写真4　後方部分の変形

写真5　さらに大胆な変形

「STELLAR ASSAULT」ゲームレビュー

いつも唐突に行われるゲームレビューですが，今回は7月号が出る時点で発売されているはずの「STELLAR ASSAULT」を紹介します。ただしこのゲーム，X68000用ではありません。セガのスーパー32X用です。では，なぜこのゲームを取り上げるかといえば，制作したのがあの「STAR WARS」の土田さんです。そうです，「STAR WARS」ではワイヤーフレームでしたが，今回はポリゴンであいかわらず“CG”しています。ということで，サンプル基板をもって遊びに来た，土田さんにいろいろ伺ってみましょう。

＊　＊　＊

かまた(以下か)：まず，このゲームを簡単に紹介していただけますか。

土田(以下土)：へい，どうも土田です。このゲームは極めてオーソドックスな3Dシューティングです。一応マニアの方から初心者の方まで遊べるような体裁は整っていますので，どなたも安心してご購入ください(すかさず宣伝)。

か：例によって，リプレイモードが楽しいですね。今回は，ちゃんとポリゴンしているので，このままCGAコンテストの作品になりそう。

土：そうですね。戦艦をかすめるように飛行すると，リプレイ時に(戦艦が)大きく表示されてナカナカに迫力が出ます。あ，あんまりギリギリばっかり飛ばないでください。ソートのボロが出ますから……。

か：2面目の重力カタパルトを破壊する面なんか，カタパルトの巨大感が出てていいですよね。うまくカタパルトに突入すると，加速するのもスピード感があって気持ちいいです。

土：僕は4面目の対艦隊戦が好きなんですよ。戦艦が10隻も登場しますから，それらをバンバン撃沈して破壊の快感を味わうのがたまりません。

か：戦艦や戦闘機など，かなりの種類が出てきますね。

土：戦闘機8種，戦艦5種，その他イロイロとキャラクタは結構出てきますね。戦艦1隻に100ファイル以上使ったものもあり，僕だけでも1000ファイル以上制作しました。死ぬかと思いましたよ。デザインはアニメなどで有名なアートミックさんにお願いしました。

か：ゲームのデモもなかなかいいですけど，あれも土田さんが作ったのですか？

土：デモ用のリプレイ映像のことですね。ハイあれは僕のプレイですよ。何度か遊んでみて見栄えがいいのを使いました。リプレイモードは，最長40分以上可能です。

か：BGVにもなりますね。ぜひ，店頭でデモをしているのを探してみてください。

＊　＊　＊

ということで，このゲームで遊び，リプレイモードを録画して，かっこいいカットを集め，「GENIE」で作り直せば，宇宙バトルCGA作品が完成します。スーパー32Xをお持ちの方には，戦闘シーンの研究用に，お勧めのゲームです。

で変形させた結果です。また，写真5は，同様の方法で，もっと大胆に変形させた例です。モデリングの参考にしてください。

アニメーション

BOXTRANS.X は，モデリングに使えるだけでなく，ちゃんとアニメーションもできます。使い方は EXPOINT.X と基本的に同じです。

/S1,20,3

のようなオプションをつけると，1フレーム目から20フレーム目までの20段階で変形が行われていくときの3フレーム目を出力してくれます。ですから，このオプションをRENCON.Xといっしょに使用すれば，少しずつ変形していくアニメーションができます。EXPOINT.Xの「/F」オプションと同じです(ならば同じオプションに統一すればいいのに)。

RENCON.X のコマンドファイルの例をリスト2に紹介します。

ただし，EXPOINT.Xでは「div」を使うことで，10フレームはこの位置に移動して，20フレームはここに移動するといった指定が可能でしたが，BOXTRANS.Xにはその機能はありません。最初の状態(直方体領域)と最終的な変形を指定するだけです。

それではこのBOXTRANS.Xを使って，どのようなアニメーションができるか考えてみましょう。まず基本的な使い方の例として，写真6-A～Dのようなアニメーションを作ってみました。

写真6-Aが元々の形状，写真6-B，Cが頭部を変形させたもの，そして写真6-Dでは，頭部の上部と後部の上部のローター接続部を2回に分けて変形したものです。写真6-Bの下を見ている変形のコマンドファイルがリスト3です。

やってみた感想としては，RENCON.Xを使っているので，作画に時間がかかり，また事前にWIREVIEW.Xで動きを確認できないなどの問題はあるものの，思ったほど制御は難しくなく，なんのトラブルもなくすんなり制作することができました。少なくともEXPOINT.Xよりずっと実用性はあります。

できたアニメーションは，プニュプニュした動きがなかなか目新しく，従来にはない作品ができるような気がします。

アニメーションのテクニック

この写真6-A～Dですが，まず頭部のプロペラは別パーツになっており，変形したあとでくっつけています。このように，変形したくない部分は別パーツにしておくというのもテクニックのひとつといえるでしょう。BOXTRANS.Xの変形は，基本的に1次補間(直線補間)ですから，プロペラも最初の位置と最終的な位置と向きがわかれば，それを「div」で1次補間してやれば，変形に合わせて移動するはずです。

次に，写真6-A，Bのように首を曲げるときの注意ですが，角度としては，45度くらいが限界です。それは，BOXTRANS.Xの原理から考えても当然で，曲がっているというよりは，歪ませているだけです。写真6-Bなどはかなり無理をしており，この方向からはわかりにくいのですが，上から見ると，少し不自然になっています。

それから，このアニメーションでは，写真6-Aの状態から写真6-Bの状態に動いた後，写真6-Bの状態から一度写真6-Aの状態に戻って，それから写真6-Cの状態にしています。なぜなら，写真6-Bの状態から直接写真6-Cに変形する方法がないからです。

BOXTRANS.Xは，直方体領域をある形状に変形する過程を少しずつ行ってアニメーションすることしかできません。つまり，最初は必ず直方体の状態でないといけ

リスト2　RENCON.X のコマンドファイル

```
#frame ( fno , 1, 20 )
    rendargs = BOX.SUF *.ATR CHPRO.SUF F1.FSC /a2 /g /hBACK.PIC /
s$fno$:$fno$
    BOXTRANS ch47 /obox /fBOX.TXT /s1,20,$fno$
```

リスト3　写真6-Bのコマンドファイル

```
box( 950 160 240   520 -160 -210 )
vector(
        200     0       -350
        0       0       0
        0       0       0
        200     0       -350

        -250    0       -100
        0       0       0
        0       0       0
        -250    0       -100
```

A　変形前の形状

B　頭部を変形

C　頭部の変形

D　頭部と後部を2回に分けて変形

写真6

ないわけです。これは仕様的な限界といえます。

しかし，限界といえどもない知恵を絞ればなんとかなります。まず安易な解決策としては，一度変形させた形状に対して，もう一度BOXTRANS.Xを実行する方法があります。けれども，この方法では変形による誤差が積算されますし，変形後の頭部は直方体領域では指定しにくい形状になっていることが考えられます。

もうひとつは，BOXTRANS.Xのアニメーションの機能に頼らず，補間を自分で行う方法です。この場合，フレーム数と同じ数のコマンドファイルを用意する必要があります。そんなの実用性がないと思うかもしれませんが，FF.Xを利用すれば，大量のコマンドファイルを手軽に生成することができます。

リスト4をごらんください。これをFF.Xで実行すると，補間した値のコマンドファイルが大量に作成されます。FF.Xには，こういった使い方もあるんですね。しかし，この出力ファイルは，大量のコマンドファイルが1つのファイルの中に全部入ってしまいます。そこで，さらにトリッキーな技として，マニュアルにも載っていない隠しオプション「/S」を使い(使い方はREND.Xの「/S」オプションと同じです)，特定のフレームだけを出力させます。

さらに，これをRENCON.Xのコマンドファイルの中に記述すれば，一発で全部やってくれます。このコマンドファイルをリスト5に載せます。注意する点は，FF.Xの出力ファイル名の指定が「/O」オプションではないという点と，BOXTRANS.Xには補間をさせないので「/S」オプションはいらないという点です。

などという説明を長々とされてもさっぱりわからないでしょう。とりあえず，読み飛ばしておけばよいのですが，BOXTRANS.Xを本格的に使うときは必ず役に立つテクニックです。そのときに“そういえば，なんか高度な使い方がOh!Xに書いてあったなぁ”と思い出してください。そして，先ほどのリストを参考にして試行錯誤すれば，きっとわかっていただけると思います。

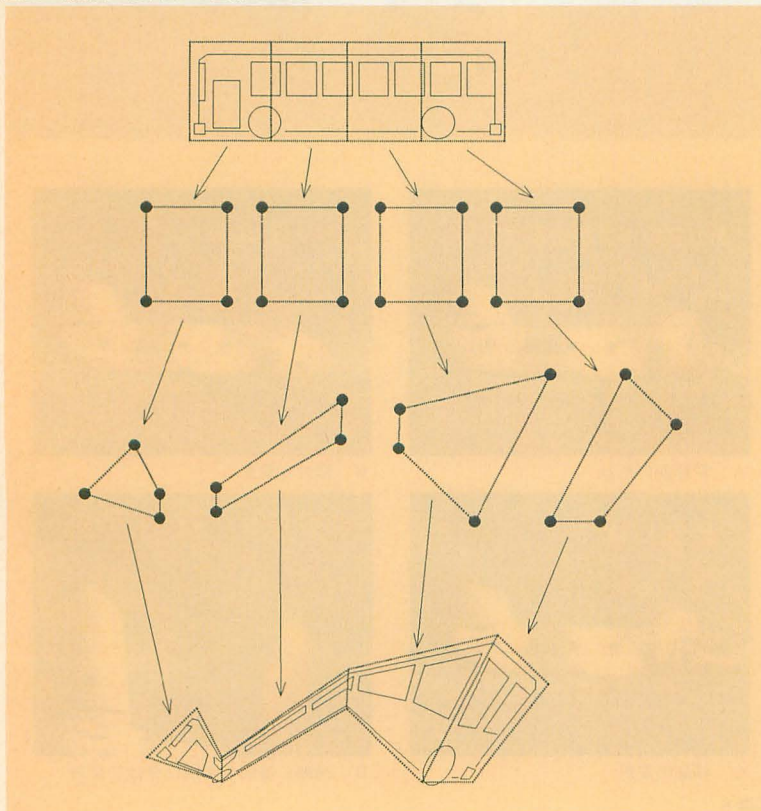
リスト4　FF.Xに通すコマンドファイル(BOX.TXT)

```
#frame( fno, 1, 10 )
@5.0@
box( 950 160 240  520 -160 -210 )
vector(
        ¥div (  200, -300, 1, 10, fno )¥
        ¥div (    0,  150, 1, 10, fno )¥
        ¥div ( -350,   50, 1, 10, fno )¥
        0       0       0
        0       0       0
        200
        ¥div (    0,  250, 1, 10, fno )¥
        ¥div ( -350,  -50, 1, 10, fno )¥

        ¥div ( -250, -300, 1, 10, fno )¥
        ¥div (    0,  150, 1, 10, fno )¥
        ¥div ( -100,   80, 1, 10, fno )¥
        0       0       0
        0       0       0
        ¥div ( -250,  200, 1, 10, fno )¥
        ¥div (    0,  250, 1, 10, fno )¥
        ¥div ( -100,  -20, 1, 10, fno )¥
)
#endframe
```

リスト5　FF.Xを含めたRENCON.Xのコマンドファイル

```
    #frame ( fno , 1, 10 )
    rendargs = BOX.SUF *.ATR CHPRO.SUF F1.FSC /a2 /g /hBACK.PIC /
of3 /s$fno$:$fno$
    FF BOX.TXT BOX2.TXT /s$fno$:$fno$
    BOXTRANS ch47body /obox /fBOX2.TXT
```

図9　複数に分割して変形させる

終わりに

現在，当チームではコンテストのビデオの発送作業や来年のためのファイルの整理をまだ行っています。さらに今年は新入生への教育に力を入れていますので，あいかわらずバタバタと忙しい毎日です。

ということで，BOXTRANS.Xももっといろいろ試したいのですが，時間が足りなくなってしまいました。すみません。

基本的なテクニックは，ひと通り述べたつもりですので，これらを組み合わせることで，さらに大胆な変形に挑戦してみてください。たとえば，図9のように，ZANTE.Xで複数の形状に分割し，それぞれをBOXTRANS.Xで変形してくっつけることで，かなり自由度の高い変形になるはずです(雑巾を絞るような変形も可能)。

もちろん，BOXTRANS.Xでも絶対に不可能な変形も多くありますが，京大マイコンクラブが制作した「MOUSE」や，努力すればディズニーアニメのキャラクターのような動きも可能でしょう。CGAコンテストにおいてもまだほとんど使われていないテクニックですので，プニュプニュした動きは目を引くこと間違いなしです。手間はかかるものの，挑戦する価値はあると思います。

さて次回は，いよいよYAWARA.Xで恐竜をという話もあったのですが，やはり変形関係は難しすぎて，多くの読者がついて来れないという問題もあります。もう少し，一般的な話をしてほしいという依頼もきていますので，ちょっと考えておきます。では，また。

[特集]

Optimizing Method

「最適化」。速度を速くすることを指す場合が多いが，速度を速くする以外にも，メモリの消費量を減らしたり，ファイルサイズを小さくしたり，ディスクアクセスを少なくしたりといったことも含まれる。基本は無駄をなくし洗練されたプログラムにすることである。速度的な最適化という概念が一般に通用しているということは，普段我々はCPUあるいはコンピュータの能力をあまり引き出せていないということを意味する。プログラムを最適化されてもコンピュータは速くなったりしない。もともとそれだけの速度を持っていたのである。

最近は，血道をあげてプログラムを最適化するよりは速いマシンを使えばいい，という考え方になりがちな人が多い。確かにAT互換機のゲームなどを見ると，すでに「Pentium Reqire」といったものも少なくないし，確かにそれだけの処理を行っているものもある。しかし，なんでもかんでもその理屈で片付けるのは，外食産業が発達しているので料理を覚えることには意味がない……という考え方と同じだ。これは単に価値観の問題ではない。最適化という行為は，結果的に速度やメモリ効率といったものに表れてくるとはいえ，その根底にあるものは「よりよいものを作ろうとする心」にほかならないのだ。

さて，こういった「最適化」という論点は，すでに「完成されたプログラムが作成できる」ことが第一条件になっている。物事を達成する能力があって初めて意味をなす。最適化という美しい言葉に酔って，全体は未完成のまま，部分だけの最適化を繰り返すなどは愚の骨頂といえる。ひとつのものを作り上げることに比べれば，公開された手順で速度や効率を上げることはさして難しいことではない。

その上で，さらに高い完成度を目指すこと。

優れたハードウェアが鬼に金棒であるか豚に真珠であるかは使い方ひとつの問題となる。

CONTENTS

中級プログラマに贈る

一線を超えた68系プログラマ養成講座

Nishikawa Zenji 西川 善司

アセンブラレベルで扱いやすい命令体系を持つCPU68000
X68000でマシン語を使うことはなかば宿命といえるだろう
まずはアセンブラでプログラムを作成する際の定石とテクニックを見てみよう

はじめに

アセンブラプログラミングの魅力とは以下の2つが大きいと思う。

1) パズル性
2) コンピュータ支配感覚

1)について。アセンブラプログラミングでは高級言語によるプログラミングとは違い，使えるアドレッシング，レジスタ数(変数の個数)が限られている。これがその難しさであり，また楽しさでもある。アセンブラプログラマならここまで読んで「うんうん」とうなずきながらにんまりしているはずである。この制限をいかに打開しつつ自分の思考を再現できるか。これが実に解けそうで解けない，あるいは解けなそうで解けてしまう面白いパズルなのだ。

2)について。アセンブラプログラミングの楽しさは，プログラマがプログラムを掌握できるところにもある。高級言語でプログラムしても，最終的にいったいどんなコードになっているかは通常のユーザーには計り知れない謎の部分である。コンピュータに指令しているのは確かに自分なのだが実際に動かしているのは自分のプログラムではない。その点，アセンブラで組めば実行速度が速いも遅いも100％自分の責任である。そのシステムの支配感が快感なのである。

さて，ここまで書いてきていうのもなんだが，ここでは別にアセンブラプログラミングの存亡について議論するつもりはない。ここではMC680x0に「土着」した私が自分なりに築き用いたテクニック，68系では常套手段といわれている必須テクを紹介していく。

この原稿が皆さんのプログラミングパズルの解法の助けに，そして新たなる想像のインスピレーションのひとつにでもなれば幸いだ。

機械語プログラムはドケチとなれ

巨大なプログラムの場合，ちょっとしたプログラムの無駄の積み重なりが，そのまま動作時の見た目の遅さ/速さに影響してくるものである。特に単位時間当たりに大量のデータを操作する場合は，ある特定の命令列が何回も実行されることになり，プログラム中のちょっとした無駄を省くだけでかなりプログラム速度の向上を達成できることがある。つまり，アルゴリズム＝本質をそのままに，プログラムを高速化するにはその命令レベルの最適化が高速化へのいちばんの近道であるといえる。

この命令レベルの最適化とは，

1) 目的機能を実現する速い命令の選択
2) 無駄が少なくなるような命令列の実現

のことであり，ここで話題のメインディッシュとして取り上げるものである。

なお，文中ではMC68000命令の実行クロック，命令語長などについて平然と引用しているので，本誌2月号62ページの一覧やMC68000インストラクションマニュアルなどを参照しながら読んでいただきたい。

ワーク設計法

本来ならば取り扱うデータがすべてレジスタ上で管理できればいいのだが，レジスタの数は限られており，しばらく使わないものはメインメモリ上の任意の場所へ保管しておく場合が多い。こういった退避場所を広義には「ワーク」と呼ぶ。大きなプログラムであればあるほどこのワークに対するアクセスは頻繁になり，この部分の高速化/最適化はやはりプログラムの高速化へとつながっていく。ということで最初にワークに関係したテクニックをいくつか紹介していこう。

相対アドレッシングによるワークアクセス

ラベルLABELで示される領域に格納されたデータをレジスタd0に読み出す場合は，なにも考えなければ，

```
move.b  LABEL,d0
```

となる。もしこのLABELがプログラム領域から－32768～＋32767バイトのレンジ内に存在するならば，

```
move.b  LABEL(pc),d0
```

としたほうが高速である。

MC68000では，

命令	クロック数
move.b LABEL,d0	16
move.w LABEL,d0	16
move.l LABEL,d0	20
move.b LABEL(pc),d0	12
move.w LABEL(pc),d0	12
move.l LABEL(pc),d0	16

となっており(pc)のディスプレースメント付きプログラムカウンタ相対(以下PC相対アドレッシングと略する)のほうが(pc)をつけない絶対アドレッシングよりも4クロックも速い。

(pc)というのは「PC相対アドレッシングだから相対アドレス値を計算しなきゃいけないんじゃないの？　だとしたらメンド臭いよ」と思ってしまう方もいるかもしれないが，そういった計算はアセンブラがアセンブル時に自動的にしてくれるのでプログラマはまったくアドレッシングの違いを気にしなくていい。単純に，普通に書いていた目的のラベル名の後ろに(pc)を付け足すだけでいいのだ。

相対ジャンプ「bra」や相対サブルーチンコール「bsr」のオペランド*1である飛び先のラベルは(pc)を設定しない。これはこれらの命令がPC相対アドレッシング専用

の命令だからである。bra,bsrなどでは後ろに書いたラベル名の後ろには(pc)が省略されているということになろうか。しかしmove命令などのソースオペランドはあらゆるアドレッシングを指定できるのでラベルには明示的に(pc)を指定しないとPC相対アドレッシングとして認識してもらえないのだ。

なぜPC相対アドレッシングが絶対アドレッシングよりも高速なのか。このメカニズムについて知っておこう。

MPUの動作フェーズにおけるメモリへのアクセスというものは演算などの内部処理などと比べて結構時間のかかる処理である。マシン語は実行の際，MPUはメモリから命令コードを読みにいくが，これは一般のデータをメモリから読み出す手間と同じである。このマシン語命令コードの長さはアドレッシングによって長くなったり短くなったりする。例外はあるが一般的に同機能を実現する場合は命令語長が短くなるマシン語を選択したほうが高速になる。

たとえば先ほどの例について見てみよう。

```
move.b  LABEL,d0
```

のマシン語列は，

```
2039 XXXX XXXX
```

(XXXX XXXXはLABELの32ビット絶対アドレス)

で語長は6バイトである。一方，

```
move.b  LABEL(pc),d0
```

は，

```
203A XXXX
```

(XXXXはLABELまでの16ビット相対アドレス)

で，語長は4バイトである。後者のほうが2バイト命令語が短い。

LABELというワークの内容のデータをd0.bへ転送するという機能自体は違わないのに後者のほうが4クロックも高速なのは，実行時に命令自体を読み出す作業が2バイト分少ないからである。

どうして後者のほうが2バイト小さいかは，そのアドレッシングの持つ情報量を考えれば一目瞭然である。

```
move.b  LABEL,d0
```

ではMPUが扱うことのできる32ビット論理アドレスの値そのもの(32ビット＝4バイト)がマシン語コードの中に含まれる。これに対して，

```
move.b  LABEL(pc),d0
```

ではこの命令が存在するアドレスからLABELというワークアドレスまでの16ビット範囲で表される距離(16ビット＝2バイト)をマシン語コードの中に含む。つまり，2つの命令語長の差はこのLABELを指し示す情報量の差そのものだったのである。

＊1　オペランド:アセンブラのアセンブリ言語における命令語が取りうるパラメータのこと

ショートアドレッシングによるワークアクセス

さて，[チャート1]はなにもPC相対アドレッシングに限ったことではない。ここではその他の応用例を見ていこう。

Human68k ver.3.XXでは共通ワークエリアであるアドレス$00000CBCにいまHuman68kが動作しているマシンに搭載されているMPUの種類を表す値が格納されていることになっている。その値と搭載MPUとの関係は以下のとおりである。

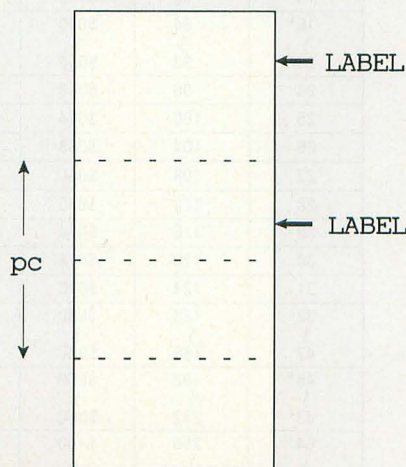
00000CBCの内容	MPUの種類
0	MPU68000
3	MPU68030
4	MPU68040

この$0000CBCはスーパーバイザ領域であるためユーザーモードからは参照できない領域であるが，スーパーバイザモードで動作するようなプログラムであるならば直接的に，あたかも自分のプログラム領域であるかのように自由に参照できるようになる(暴走時にシステムダウンの危険性は伴うが)。

その際，たとえばMPUが68030であるかどうか試験する場合，

```
cmpi.b    #$03,$00000cbc
beq       MPU_MC68030
```

といった記述をするのが普通であろう。

ところがMC680x0ではアドレス$00000000を中心とした$FFFF8000〜$00007FFFの範囲は特別にショートアドレッシングというアドレス指定のしかたが許されている。これを適用すると上記の記述は，

```
cmpi.b    #$03,$0cbc.w
```

となる。

これはアドレスを16ビット長で表し，実際にそのアドレスに対してなんらかのアクセスをする場合にはこれを32ビットに符号拡張してから行う，というものだ。

通常OSの共通ワークやI/Oアドレスなどは論理アドレスの最上位ブロックや最下位ブロックに割り当てられているコンピュータが多い。ハードやOSに密着したプログラムを作成する場合，こういった領域に頻繁にアクセスするのでそういった場合を想定して設けられたアドレッシングであるといえよう。

```
cmpi.b    #$03,$00000cbc
```

より，

```
cmpi.b    #$03,$0cbc.w
```

が高速なのはなぜかはもうおわかりだろう。

そう，前者は第2オペランド'$00000cbc'が32ビット(＝4バイト)，後者は'$0cbc'で16ビット(＝2バイト)であり2バイト命令語長が短くなるため，その分高速となるのである。

チャート1

命令語長が短くなるアドレッシングを用いよ
(＝命令語長が短くなるように適切な命令を選択する)。

例.
絶対アドレッシングを相対アドレッシングにする。

```
move.b  LABEL,d0
   ↓
move.b  LABEL(pc),d0
```

図1　相対アドレッシングのようす

このアドレッシングはHuman68kなどの共通ワークを触る以外にも，各種割り込みベクタの書き換えなどにも有効である。MC680x0では表1のように各種割り込みベクタが下位アドレスにまとまってマッピングされている。

たとえばデバイスドライバなどを作成する場合，TRAP命令($00000080〜$000000BC)やその他，ハードウェア割り込み($00000100〜$000003FF)のベクタを書き換えることはよくあることである。

ベースアドレスを設定しよう

ショートアドレッシングやPC相対アドレッシングはそれらはそれらで便利だがそれなりの欠点もはらんでいる。まず，ショートアドレッシングは論理アドレスの最上位や最下位の領域に限ってしか使用できない。PC相対アドレッシングはアクセス元から16ビットレンジを超えた遠いアドレスには適用できない。

とはいってもショートアドレッシング形式は最上位/最下位領域を触るときに用いるものなのだと割り切って使えばいいし，PC相対アドレッシングについても，よほど巨大なプログラムソースを書かなければ参照したいワークが16ビットレンジを超えた先に行ってしまうことはないので，欠点という欠点はないように思える。

ところがPC相対アドレッシングにはもうひとつ重大な欠点がある。それはPC相対アドレッシングは多くの命令の第2オペランド(デスティネーションオペランド)に使用できないという制約である。その制約とは，オペランドで示されるアドレスの内容が更新される場合はPC相対アドレッシングは使用できないというものである。つまり，

```
move.b  LABEL(pc),d0
```

はできても，

```
move.b  d0,LABEL(pc)
```

は許されないのだ(アセンブルエラーが発生する)。この制約の理由はよくわからないうえ，例外もある。データテスト命令であるtstや比較命令cmpiはオペランドの内容を更新する命令でないのに，

```
tst.b   LABEL(pc)
cmpi.w  #$1234,LABEL(pc)
```

といった表記は許されないのである*1。同じような命令でデータのビットテストを行うbtst命令については，

```
btst.b  #7,LABEL(pc)
```

のように許されるのであるから，思わず混乱してしまいそうだ。

実際ホンのちょびっとしか離れていない領域を指し示すのに4バイトものアドレス値がマシン語コードに含まれるなんていうのは実に無駄なような気がしてならない。なるべく32ビットアドレッシングを使わないで済ませないものか。

このほか，32ビットを使わないアドレッシングにはディスプレースメント付きアドレスレジスタ間接というものがある(以下ディスプレースメントとオフセットは同義とする)。

なんと，この方式は全命令の第1オペランド(ソースオペランド)はもちろん第2オペランド(デスティネーションオペランド)に対しても使用でき，さらにPC相対アドレッシングでは許されなかった第2オペランドの内容が更新されるような場合においても使用できるという万能なアドレッシングである。速度的にもPC相対アドレッシングとまったく同等である。なんとかこれを使う手はないか。

ところでアセンブラプログラミングでは明確にプログラム部(コード部)とデータ部を分けることが多い。このデータ部のワークは，画像データや音声データのような大容量データそのものを格納する領域としてではなく，そういった大容量データの存在するアドレスやデータサイズ，その他各処理に必要なパラメータ，定数などを格納する目的で割り当てられているのが普通である。

こういったワークは各ルーチンの直後に局所的に設定される場合もあるが，ほかのルーチンから共有する形でちょくちょく参照するタイプのデータはやはりプログラム中の決まった領域にまとまって割り振られているものである。

そこでこういったワークのおおよそ中心部にベース(基準)となるラベルを設定し(なぜ中心かは後述)，このアドレスをユーザーが決めたアドレスレジスタに設定する(ベースアドレスの設定)。で，実際のワークへのアクセスはこのベースレジスタにオフセットを与えたディスプレースメント付

表1　例外ベクタの割り当て

ベクタ番号	ベクタの先頭アドレス			割り当ての内容
	10進	16進	メモリ空間	
0	0	$000	SP	リセット：SSPの初期値
1	4	$004	SP	リセット：PCの初期値
2	8	$008	SD	バスエラー
3	12	$00C	SD	アドレスエラー
4	16	$010	SD	不当命令
5	20	$014	SD	0による除算
6	24	$018	SD	CHK命令
7	28	$01C	SD	TRAPV命令
8	32	$020	SD	特権違反
9	36	$024	SD	トレース例外処理
10	40	$028	SD	未実装命令 (line 1010 emulator)
11	44	$02C	SD	未実装命令 (line 1111 emulator)
12*	48	$030	SD	割り当てられていない（予約されている）
13*	52	$034	SD	割り当てられていない（予約されている）
14*	56	$038	SD	割り当てられていない（予約されている）
15	60	$03C	SD	アンイニシャライズド割り込み
16*〜23*	64〜92	$040〜$05C	SD	割り当てられない（予約されている）
24	96	$060	SD	スプリアス割り込み
25	100	$064	SD	レベル1割り込み，オートベクタ方式
26	104	$068	SD	レベル2割り込み，オートベクタ方式
27	108	$06C	SD	レベル3割り込み，オートベクタ方式
28	112	$070	SD	レベル4割り込み，オートベクタ方式
29	116	$074	SD	レベル5割り込み，オートベクタ方式
30	120	$078	SD	レベル6割り込み，オートベクタ方式
31	124	$07C	SD	レベル7割り込み，オートベクタ方式
32〜47	128〜188	$080〜$0BC	SD	TRAP #0 命令〜TRAP #15 命令
48*〜63*	192〜252	$0C0〜$0FC	SD	割り当てられていない（予約されている）
64〜255	256〜1020	$100〜$3FC	SD	ユーザー割り込みベクタ（192種類）

＊これらのベクタ番号は将来使用される可能性があるので，ユーザが使用すべきでない。

きアドレスレジスタ間接アドレッシングで行えば，たとえ実行しているプログラムとデータ領域の距離(アドレス差)が16ビットレンジを超えた位置関係にあったとしても，参照したいワークが，前もって設定したベースアドレスから16ビットレンジ内にあればPC相対アドレッシングと同等の速度でアクセスが可能となる。

また，ディスプレースメント付きアドレスレジスタ間接アドレッシングは先述のとおり第2オペランドにも適用できるのがなんともおいしいではないか。

いくつか例を示そう。あるワークエリアが，

```
work_A:      dc.b  1
work_B:      dc.b  2
work_C:      dc.b  3
   :
   :
work_X:     dc.b  24
work_Y:     dc.b  25
work_Z:     dc.b  26
```

といったラベル名で，データサイズはすべてバイトサイズ，内容は数列1，2，3，……26が順番にそれぞれのワークに格納されているとしよう。

まず，ベースを設定する。ベースはここではおおよそワークの中心に位置するラベルwork_Nに設定することにしよう。ベースを設定するアドレスレジスタは適当にa6とする。

設定にはもちろん実行アドレスのロード命令leaを使用する。ベースアドレスの設定は，

```
lea      work_N,a6
```

となる。もちろんこのwork_Nがプログラム部から16ビット範囲内であるならば，

```
lea      work_N(pc),a6
```

と書けるのはいうまでもない。

さて，ベースを設定し終わったところで，次に「work_Fの内容とwork_Xの内容を加算しこれをwork_Xに格納する」という例を考えてみることにする。

```
move.b   work_F−work_N(a6),d0
add.b    d0,work_X−work_N(a6)
```

と，こんな感じになる。まず1行目だが，これはwork_Fから値を取り出している。もちろんPC相対アドレッシングが使用できる場合ならば，

```
move.b   work_F(pc),d0
```

でも構わない。が，PC相対アドレッシングが届かない位置にwork_Fがある場合にはこのディスプレースメント付きアドレスレジスタ間接アドレッシングが有用だ。

すでにワークの中心部分あたりのアドレスをベースとしているので，ベースから目的のワークまでの変位(オフセット)だけで目的を指し示すことができる。

次の行，

```
add.b   d0,work_X−work_N(a6)
```

ではいっそうこの方式の有用性が再認識できる。この場合，たとえwork_XがPC相対アドレッシングで指し示すことのできる距離内にあったとしても，先述した制約により，

```
add.b   d0,work_X(pc)
```

という記述は許されないからだ。

一応，

```
work_X−work_N(a6)
```

という表記の意味を説明しておこう。

ディスプレースメント付きアドレスレジスタ間接アドレッシングとはベースとなるアドレスから相対的に(この相対値の幅は16ビットレンジ)場所を示すものである。この場合ならばa6の内容である。たとえば，

```
10(a6)
```

ならばa6+10のアドレスが参照され，

```
−25(a6)
```

ならばa6−25のアドレスが参照されるということである。PC相対アドレッシングではプログラムカウンタPCを基準に「±いくつ」という計算を参照したい箇所のラベル名の後ろに(pc)を，

```
LABEL(pc)
```

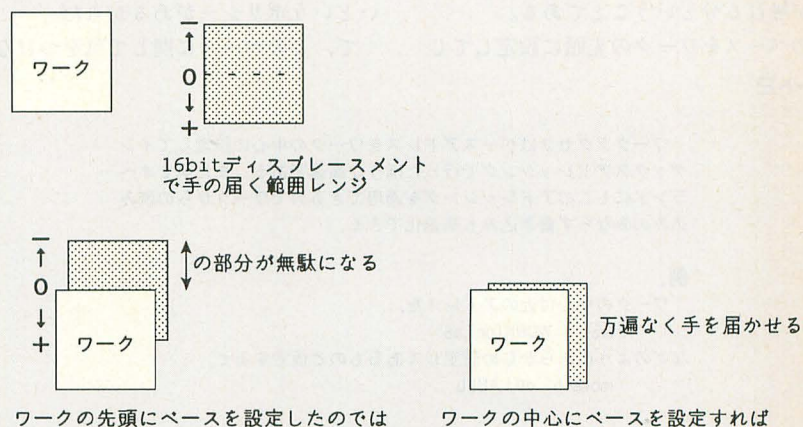
のようにつけるだけでこのオフセットを自動的に計算してくれたが，ディスプレースメント付きアドレスレジスタ間接アドレッシングでは参照先のラベル名を，

```
LABEL(a6)
```

のように書いても意味をなさない。プログラマ側でその基準から「±いくつ」という

図2　なぜベースをワーク中心に設定するか

16bitディスプレースメントで手の届く範囲レンジ

の部分が無駄になる

ワークの先頭にベースを設定したのでは

万遍なく手を届かせる

ワークの中心にベースを設定すれば

魔神語の時代

その昔，横山やすし親子が「このマシン正解やで」とパソピアを指差して笑っていた8ビット全盛時代，パソコンゲームは「オールマシン語」という言葉がひとつのステータスであった。「マシン語で組まれたゲーム＝スゴイ」という図式は当時のパソコンユーザーにとって「金鳥の夏＝日本の夏」という事実よりも常識とされていた。これはなぜか。

あまり処理速度の速くない8ビットマシン時代，高級言語で作られたプログラムは(そのコンパイラ技術の未発達もあって)「遅」かった。そこでリアルタイム性を要求されるゲームプログラムにおいては，そのターゲットハード，命令組み合わせレベルまで最適化された機械語プログラミングで速度を稼ぐ必要があったのである。とはいえ，8ビットCPUはその機能自体に制約も多く，たかがゲームとはいえ丸ごと機械語で組むのは相当な労力を要する。

いまでこそICEやデバッガが充実しているが，当時，機械語のプログラムは「実行→暴走→悩む」という原始的な開発工程を強いられた。ギチギチに最適化されたプログラムというものは保守拡張が非常にやりにくい。よかれと思って施した修正が，別のルーチンに影響を及ぼしたり，と予期せぬバグの発生につながってしまう。まぁそういうわけで多くのプログラマは高級言語(主にBASIC)でメインを組み，高速性を要求される部分をマシン語で……，という開発形態をとった。実際当時の名作ゲームといわれたものもこの形態のものが多い(信長の野望，ウォークワンetc.)。

すべてマシン語で作られたゲームに対して「よくマシン語だけでこんなゲームを作れたよな。きっとスゲー奴が作ったんだからゲームもスゲーに違いねー」的な賞賛と期待の意味を含んだ信頼の構図が発生したのであった。

＊　＊　＊

マシン語は当時の「パソコン好き」にとってヒデキやピンクレディよりも「憧れ」だったのである。

値，すなわちオフセット(の式)を書いてやらないとだめなのだ*2。

今回の例の場合，参照したい目的のアドレスはwork_Xで，ベースがa6，そしてa6にはwork_Nの実行アドレスが入っているとわかっている。そこで，

work_X－work_N

で，a6(＝work_N)からwork_Xまでの距離を計算し，この結果をディスプレースメント付きアドレスレジスタ間接アドレッシングのオフセットとする，

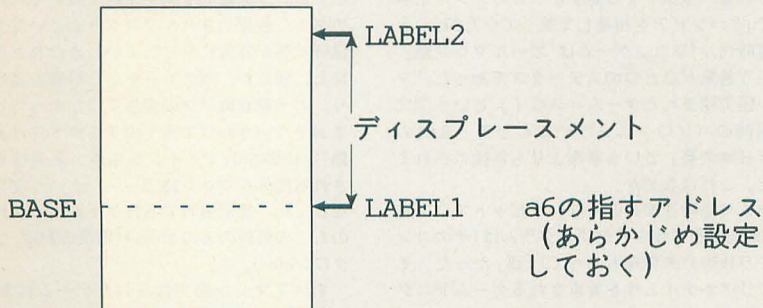
work_X－work_N(a6)

は，つまり，

a6＋(work_X－a6)

を行っていることになる。

ところで，なぜワークの中心部にベースを設定するのか。ディスプレースメント付きアドレスレジスタ間接アドレッシングでもPC相対アドレッシングと同様に，ベースとして設定したアドレス値から－32768～＋32767の範囲内の16ビットレンジ内のワークが触れる分ということである。

このベースをワークの先頭に設定してしまったのでは＋0～＋32767までの範囲(32Kバイト)しか触れなくなってしまう*3。ベースをそのワーク中心部においたのは，このアドレッシングの有用性を無駄なく発揮するために，このアドレッシングの有効範囲がもっともワークを網羅できるように設定すべきだからである(図2)。

もちろんディスプレースメント付きアドレスレジスタ間接アドレッシングによるワークアクセスもワーク自体が巨大化してしまった場合はPC相対アドレッシングと同様の限界により使えなくなってしまうが，たとえプログラムがかなり巨大になったとしても，こういった変数管理のような目的のワークが64Kバイトを超えるケースはまずないので問題はないだろう。

さて，このベース(基準)はどのレジスタに設定するか，少し悩む点である。私はベースはa6に設定することが多い。スタックのa7同様*4，グローバルな役目を負ったレジスタはレジスタ番号の大きいものにしたいというポリシーがあるからだ。

で，このベースに関して気をつけなければならない点がひとつある。ベースとなるアドレスレジスタはプログラムの先頭で設定したら絶対内容を更新してはならない，ということ。たとえばプログラム先頭で，

```
lea     work_N(pc),a6
```

を実行してあり，a6＝work_Nであるときに，

```
move.b  work_X－work_N(a6),d0
```

を実行した場合はちゃんとwork_Xの内容が参照されるが，なんらかの不手際でa6の内容がほかの値になってしまったとしよう。そこで，

```
move.b  work_X－work_N(a6),d0
```

を実行しても，

(work_X－work_N)＋??

となりまったく違った場所を指し示してしまう。非常に基本的なことだがぜひ注意したい。

*1 MC68030では許されるように拡張された。
*2 なぜPC相対アドレッシングだと参照先のラベル名を書くだけで計算してもらえるかというとPC相対アドレッシングはPC(プログラムカウンタ)が基準ということがわかっているからである。

```
move.b  LABEL(pc),d0
```

のLABEL(pc)という式を評価する場合，基準となるのは実行時のPCであるが，これは「move.b」のマシン語コードが生成されるアドレスにほかならない。であるからLABEL(pc)は，

LABELが存在する実行アドレス
　－move.bが生成される実行アドレス

という計算をアセンブラは自動的に行うのである。ではディスプレースメント付きアドレスレジスタ間接アドレッシングにおいて，たとえば，

LABEL(a6)

と書いてあってもアセンブラがアセンブルするときにこの式を評価する場合，アセンブラはa6の内容は知るよしもない。a6にはどんな値が入っているかはプログラマしかわからない(あるいはプログラムを実行してみなければわからない)。
そういうわけでディスプレースメント付きアドレスレジスタ間接アドレッシングではプログラマ自身がこのオフセットの計算式をアセンブラに教えてやる必要があるのだ。
ちなみに，

LABEL(a6)

というような表記を行うとLABELという32ビットの実行アドレスをディスプレースメント付きアドレスレジスタ間接アドレッシングに適用したということで怒られエラーとなる。しかしMC68030ではディスプレースメント(オフセット)に32ビットが使えるようになってしまったので，アセンブルエラーが起こらずにLABELの実行アドレスがオフセットとされてしまい，意味不明なアドレスを参照するような危険なマシン語が生成されてしまう。
*3 とはいってもワークが32Kバイトを絶対超えることはないと確証があるならばワークの先頭をベースとしても構わないだろう)
*4 MC680x0ではアドレスレジスタ7番のa7を特別にスタックポインタとして割り当てている。これ以外のd0～d7，a0～a6は基本的には特別な機能はない。

チャート2

ワークアクセスはベースアドレスをワークの中心に設定してインデックスアドレッシングで行ったほうが高速である。また第2オペランドにもこのアドレッシングを適用できるのでワークからの読み込みのみならず書き込みも高速化できる。

例.
ワークの中心付近のアドレスを，

```
lea     WORK(pc),a6
```

などのようにあらかじめ設定してあるものと仮定すると，

```
move.b  d0,LABEL
   ↓
move.b  d0,LABEL－WORK(a6)
```

のようにすることができる。

図3　ディスプレースメント付きアドレスレジスタ間接アドレッシング

テーブル設計法

アセンブラプログラミングにおいて「テーブル」という概念は頻出する。ざっと挙げるだけでも，ジャンプテーブル，配列，変換テーブルなどなど。ここではこれらについてのいくつかのテクニックを紹介していく。

テーブルアクセスあれこれ

ディスプレースメント付きアドレスレジスタ間接アドレッシング方式はベースアドレスを決めてこれにオフセットを加算して算出されたアドレスに対してアクセスを行うものであった。

先に紹介した方法はどちらかといえば裏技的な使い方で，本来は，ディスプレースメント(オフセット)をラベル定義しておき，この値を用いてディスプレースメント付きアドレスレジスタ間接アドレッシングを適用するのが一般的な使い方である。いわゆるテーブルへのアクセスとか，配列データへのアクセスといった話である。

実に一般的な話なのでいまさら説明はいらないかもしれないが，ここではそのテーブルや配列のアクセスについて触れてみたい。ここで取り上げるのは先出のディスプレースメント付きアドレスレジスタ間接にさらにレジスタインデックスをも指定できるアドレッシングである「インデックス付きアドレスレジスタ間接アドレッシング」だ。

ちょっと実戦的な例を挙げてみることにする。

小規模のテーブルワークの場合を考えてみよう。たとえば10個の物体の3次元座標X,Y,Zを管理する必要が出てきたとする。この座標X,Y,Zはそれぞれワード(2バイト)サイズであるとすると，ひとつの物体については2×3＝6バイトのメモリ領域が必要になる。物体10個について管理するには6×10＝60バイトの領域が必要になる。なにも考えないならば，

```
X0:  ds.w  1
Y0:  ds.w  1
Z0:  ds.w  1
X1:  ds.w  1
Y1:  ds.w  1
Z1:  ds.w  1
X2:  ds.w  1
Y2:  ds.w  1
Z2:  ds.w  1
     :
     :
X9:  ds.w  1
Y9:  ds.w  1
Z9:  ds.w  1
```

というようなワークを設定してしまうところだ。物体4のX座標を取り出す場合には，

```
move.b    X4(pc),d0
```

というふうにするわけだ。

まぁこれでも構わないが，こういうデータの場合，物体番号(0～9)を与えられてこれをキーとして目的の物体のX,Y,Zの値を参照したい状況がよく起こる。

あらかじめ，

```
X:          equ    0
Y:          equ    2
Z:          equ    4
WORK_SIZE:  equ    6
```

(ラベル1)

というラベルを定義しておく。ワークエリアに先ほど求めた物体10個分のX,Y,Z座標の格納領域である60バイトを確保しておく。これをここでは，

```
LOCATION:  ds.b    60
```

としよう。アドレスレジスタa0にはプログラム先頭で，

```
lea   LOCATION(pc),a0
```

としてa0＝LOCATIONのベースを設定したとする。ここまでで基本設定は終了だ。

それでは実際のアクセスを行うことにしよう。ある局面で物体番号nのZ座標を1増やす必要が出てきた。いま，物体番号「n」はd0に格納されているとする。

さて，物体nはワークLOCATIONのどこから管理されているかを考えなければならない。物体0は，LOCATION＋0のアドレスからX,Y,Zという順番でワードサイズの領域が割り振られている。

アドレス	ワーク名	サイズ
LOCATION＋0	物体0のX	2バイト
LOCATION＋2	物体0のY	2バイト
LOCATION＋4	物体0のZ	2バイト

ということは次の物体1は物体0のZの次，つまりLOCATION＋6から始まる。物体2はLOCATION＋12から……以下同様。

まーここまで丁寧にやる必要はないかもしれなかったが，物体番号n(0～9)を6倍すればアドレスLOCATIONから何バイト離れたところから目的の座標ワークが存在するかのオフセットが求まる。

レジスタの値の6倍という処理は，

```
mulu        #6,d0
```

という掛け算命令を用いることにする(乗算は68000という高機能MPUであっても実行時間のかかる遅い命令であるので本来は用いるべきでない。この話は後述する)。

ここまででLOCATION＋物体番号n×6とすれば物体nの座標Xがアクセスできるのはわかるだろう。さてここでは物体nの座標Zをアクセスしたいのであるから，

LOCATION＋物体番号n×6

に目的の座標名までのオフセットをさらに加える必要がある。これは初めに定義したラベル1を使う。もちろんこのオフセットはその都度，命令語に0,2,4などの数値を与えてもいいが，今回の例のようにラベル化しておけばプログラムの可読性は向上する。

ここまでを整理するとベースアドレスは，

a0＝LOCATION

d0は初め物体番号(0～9)が入っていたが6倍したので，

d0＝物体番号×6

目的の座標Zはラベル1によれば，

Z＝4

つまり「物体nの座標Z」はこれらをすべて足し合わせたアドレス，

a0＋d0＋Z

である。

実はこのような複雑なアドレス表現をMC68000ではひとつのアドレッシングで表現できてしまうのだ。これがインデックス付きアドレスレジスタ間接と呼ばれるアドレッシングである。これを用いて表現すると，

a0＋d0＋Z

は，

Z(a0,d0.w)

となる。なにも難しいことはなくツラツラと加算したい要素を書き並べるだけだ。

ここまでの処理をまとめてアセンブリ言語で書いてみよう。

```
lea      LOCATION(pc),a0
         :
         :
mulu     #6,d0
addq.w   #1,Z(a0,d0.w)
```

となる。

さてこのままではいくつかプログラムの保守上問題があるのでそれらの点について順番に言及していこう。

```
mulu        #6,d0
```

だが，これはX,Y,Z3つの座標ワークの総サイズに相当するがプログラムをさっと見ただけではなんだかわからないので，ラベル1にもあるが，

```
WORK_SIZE:  equ     6
```

のように1ワークにおけるサイズを適当なラベル名で定義しておき,

```
mulu    #WORK_SIZE,d0
```

ようにすべきである。ラベル1のように各ワークのオフセット値リストの近くにサイズなどの付随情報も記載しておくと万が一,データの構成に変更が起こった場合でもプログラム側の改造の手間が少なく済むはずだ。

「データの構成が変わる」という話が出たがラベル1のような書式は実はデータ構成が頻繁に変わるときにはプログラムの保守には不都合な書式といえる。たとえばこの座標X,Y,Zの前に物体の各座標軸に対する回転角のデータRX,RY,RZを挿入したとしよう。データの要素数が変化したのだから当然ワークのサイズも変更する必要がある。

```
LOCATION:   ds.b    60
```

だったのを,

```
LOCATION:   ds.b    120
```

にしなければばならない。

今後もデータ構成を改変する可能性があるのだとすればこのワークサイズを直値で持つのはかなり無意味である。そこでここはせっかく1物体におけるワークのサイズがラベルWORK_SIZEに定義してあるのだからこれを用いて,

```
LOCATION: ds.b 10*WORK_SIZE
```

とするのがよいだろう。

もっとも物体数10という値も変化する可能性があるならばこれもラベル定義したほうがよいだろう。するとラベル1は,

```
RX:         equ   0
RY:         equ   2
RZ:         equ   4
X:          equ   6
Y:          equ   8
Z:          equ   10
WORK_SIZE:  equ   12
N_OF_OBJ:   equ   10
```

(ラベル2)

のようにすべきだろう。リストのほうは,

```
    lea     LOCATION(pc),a0
            :
            :
    mulu    #WORK_SIZE,d0
    addq.w  #1,Z(a0,d0.w)
            :
            :
LOCATION:
    ds.b  WORK_SIZE*N_OF_OBJ
```

のようになる。

ラベル化したことによってプログラム側の変更の手間は省けても,データ構造を改変するたびにラベル定義文のequの後ろの数値をすべて直さなければならないのは大変である。

確かに,大規模なオフセットテーブルを作成する場合や開発途中などで頻繁にデータ構造を変更する場合などではequ文でオフセット値を列記していくのは効率が悪い。そこでそういう場合には疑似命令.offsetを使ってラベルを作成すると大変効率的である。

.offset疑似命令はまさにこういったオフセット表を作るための疑似命令でこの命令のあとに記載したds文は実際に領域は確保しない。offsetセクションが終了宣言されるまで記述されたラベルに対してそのオフセットアドレス値を割り当てていく。

ラベル2をもし.offset命令で記述するならば,

```
    .offset 0
RX:         ds.w   1
RY:         ds.w   1
RZ:         ds.w   1
X:          ds.w   1
Y:          ds.w   1
Z:          ds.w   1
WORK_SIZE:
N_OF_OBJ:   equ    10
    .text
```

(ラベル3)

となる。これならばデータ構成要素がいくら追加されても,

```
LABEL:      ds.?   1
```

のようにラベル名とデータサイズを適当な位置に挿入するだけでアセンブル時にアセンブラが適切なオフセット値を各ラベルに割り当ててくれる。

ちょっとラベル3について解説しておこう。まずWORK_SIZEの後ろになにもないのはこれはワザとである。ラベルWORK_SIZEは1物体が持つワークのサイズを示す値を定義していたラベルだった。ワークのサイズというものは,

ワークの最終アドレス
　－ワークの先頭アドレス

で求められる。この場合ならば最終アドレスはZ+(Zのサイズ)すなわちZ+2である。であるからしてラベルWORK_SIZEをZの次に記述しておくことにより,

```
    :
    :
Z:  ds.w   1
WORK_SIZE:  ←Z+2のアドレスを
             アセンブラが
             割り当ててくれる
```

のようにアセンブラはWORK_SIZEにZの次のオフセットアドレスを割り当ててくれるはずである。

今度はワークの先頭アドレスだが,この場合はRXである。.offset疑似命令のパラメータはこの先頭のワークのオフセットアドレスを決定するものである。ラベル3では先頭に,

```
.offset    0
```

とあることから,つまり,

RX=0

である(もちろん.offset −16などと書かれていたらRX=−16である)。

ということでワークサイズは,

WORK_SIZE−RX

で求められるのだがRX=0のため,

WORK_SIZE−0=WORK_SIZE

ということでワークサイズはオフセットテーブルの最終アドレスそのものなのである。

ところでラベル3の最後に,

```
N_OF_OBJ:  equ  10
```

という記述があるがこの10という値は.offset文のパラメータには影響されない。

```
    .offset    −10
LABEL:      equ    11
```

アセンブラプログラミングの学習は無駄か

いかに速い命令を並べるか,これが冒頭で述べた大変面白いパズルであり,アセンブラプログラミングの魅力である。しかし,1つひとつのMPU(CPU)ごとにその生い立ちや設計者(社)の思想により(場合によっては)かなり片寄った命令系を持っている場合があり,このパズルを解くためには,そのMPUならではの高速化テクニックを取得する必要がある。これはアセンブラプログラミングの欠点でもあり,高級言語コンパイラであればこの欠点を「機械的」であるとはいえ,なんなく吸収してくれる。つまり,特に動作対象MPUのノウハウはなくても取り決められた言語仕様の知識さえあればよく,メンド臭いことは全部コンパイラが代行してくれるというわけだ。高級言語プログラミングならば無駄な知識獲得の手間を省いてプログラミングに専念できるということでもあるわけだ。

しかしなにかしらひとつのMPUについての機械語プログラミングのノウハウを学ぶと,「コンピュータの動作」という本質的な部分について理解することになり,別のMPUやハードウェアの機械語プログラミングについてもなんなく入っていける。さらに高級言語でプログラミングを行う場合ですら,より高速な実行を実現するコーディングのための発想を促してくれるはずである。

としてあってもLABELに11－10＝1が割り当てられることはない。

最後の.textは.offsetセクションの終了を宣言する意味あいで使用している。.text以外に.data,.bss,.stack,.endでも.offsetセクションは終了する。

掛け算と変換テーブル

前節で「乗算は遅い」ということを書いてしまったが，乗算は使うべきでない，という意味ではない。自前で乗算ルーチンを組んだところで1命令で内部的に演算してしまうmulu/muls命令にかなうわけがない。まぁただ掛け算命令を使うよりも明らかに別の命令で代用したほうが高速である場合は，掛け算命令を無理して使うことはないといっているのだ。

たとえば，よく知られているのは，値を2のべき乗倍する場合だ。d0.wの値を2倍するときは，

```
lsl.w     #1,d0
```

としたほうが，

```
mulu      #2,d0
```

とするよりずっと高速である。同様に，

```
4倍するならば    lsl.w  #2,d0
8倍するならば    lsl.w  #3,d0
16倍するならば   lsl.w  #4,d0
```

とできる*1。ただし，2倍，4倍ならばlsl命令よりも，

```
add.w     d0,d0
```

のほうが高速である。これは自分自身を自分自身に加算する，すなわち2倍に相当する演算である。4倍はこれを2つ並べて，

```
add.w     d0,d0
add.w     d0,d0
```

とする。これがどのくらい速いかを表にまとめてみた。バイト/ワード*2とロングワードでは命令の実行クロックが異なるので分けて比較してみることにする。実はなかなか興味深い比較結果が得られるのだ。

表2でわかるようにバイト/ワードサイズでは，4倍まではaddのほうが速いが16倍からは逆に遅くなっている。一方ロングワードでは2倍のときしかaddは速くない。これはおたっきーアセンブラプログラマを目指すならばぜひ覚えておきたい情報だ。

前節では6倍という2のべき乗では表せない倍率の乗算があった。では掛け算命令を使うしかないのだろうか。

これは微妙な質問だが，一応掛け算命令を用いるよりも速いといわれている方法がないこともない。これもよく知られている方法だと思うがワークレジスタを1個用いて乗算を2のべき算の計算と加算に分割して行うのだ。たとえば6倍ならば，

d0×6＝d0×2＋d0×4

として演算を行うのだ。つまり，

```
add.w    d0,d0   *d0*2
move.w   d0,d1   *d1=d0*2
add.w    d0,d0   *d0*4
add.w    d1,d0   *d0*4+d0*2=d0*6
```

のようにする。これ全部で実行クロックは4×4＝16クロック。6倍は，掛け算命令muluを用いた場合は46クロックであるから確かにずいぶんと速い。しかし，ワークレジスタを必要とすることが欠点である。

このような小規模な掛け算ならばほかにはテーブルを用いる方法もある。前節の例の場合掛ける数は6倍，そして掛けられる数である物体番号もたかだか0～9の10個の数値しか取りえないことはわかっている。そういうわけで起こりうる計算式をすべてあらかじめ計算しておきその結果をテーブルとしてメモリに展開しておく。そして物体番号0～9をキーとしてこのテーブルから演算をせずに結果だけもらう……という手法だ。具体的にはこんな感じになる(d0.w＝物体番号)。

```
        :
    move.b mul_rslt(pc,d0.w),d0
        :
mul_rslt:
    dc.b    0,6,12,18
    dc.b    24,30,36,42
    dc.b    48,54
```

これで14クロック，先ほどのべき算和算法よりも2クロック速い。このアドレッシングはインデックス付きPC相対と呼ばれるものですでに紹介したPC相対アドレッシングにインデックス機能が備わった究極のアドレッシングである(テーブルアクセスのインデックス付きアドレスレジスタ間接のPC版という見方もできるが)。

```
move.b     mul_rslt(pc,d0.w),d0
```

はイメージ的には，

```
mul_rslt(pc)+d0.w
```

という感じでmul_rsltのテーブルの第d0.w番目のデータをd0に取り出すという処理になる。mul_rsltは6×0,6×2,6×3,……,6×8,6×9の演算結果が順番に並べられている定数テーブルである。

つまりmul_rsltテーブルから第d0.w番目の値は6*d0.wの値が格納されているのだ。べき算和算法よりも優れているのはなにも実行速度だけではない。1命令で実行できる点*3とワークレジスタを一切使用しない点もある。ただし欠点はmul_rsltとその演算結果取り出し命令である，

```
move.b    mul_rslt(pc,d0.w),d0
```

との距離が8ビットレンジ(－128～＋127)の範囲内になければならないという制約があることだ*4。

また大規模な乗算になるとテーブルがやたらメモリを食うようになりあまり現実的な手法ではなくなってくる。しかし，演算速度が高速であるためきわめて厳しい速度要求がなされているときにはメモリを犠牲にしてでもこの手法が用いられることがある。

*1 lsrを使えば逆に値の1/2倍，1/4倍という1/2nの演算が行える。
例) d0の値を1/2倍する
```
lsr.w  #1,d0
```
ただし，値が符号付き整数である場合ならばasrを使う必要がある。

表2

バイト/ワードサイズの場合(lsl)

倍率	命令	実行クロック
2倍	lsl.w #1,d0	8
4倍	lsl.w #2,d0	10
8倍	lsl.w #3,d0	12
16倍	lsl.w #4,d0	14
32倍	lsl.w #5,d0	16
64倍	lsl.w #6,d0	18
128倍	lsl.w #7,d0	20
256倍	lsl.w #8,d0	22

バイト/ワードサイズの場合(add)

倍率	命令	実行クロック
2倍	add.w d0,d0×1	4
4倍	add.w d0,d0×2	8
8倍	add.w d0,d0×3	12
16倍	add.w d0,d0×4	16
32倍	add.w d0,d0×5	18
64倍	add.w d0,d0×6	20
128倍	add.w d0,d0×7	22
256倍	add.w d0,d0×8	24

ロングワードサイズの場合(lsl)

倍率	命令	実行クロック
2倍	lsl.l #1,d0	10
4倍	lsl.l #2,d0	12
8倍	lsl.l #3,d0	14
16倍	lsl.l #4,d0	16
32倍	lsl.l #5,d0	18
64倍	lsl.l #6,d0	20
128倍	lsl.l #7,d0	22
256倍	lsl.l #8,d0	24

ロングワードサイズの場合(add)

倍率	命令	実行クロック
2倍	add.l d0,d0×1	8
4倍	add.l d0,d0×2	16
8倍	add.l d0,d0×3	24
16倍	add.l d0,d0×4	32
32倍	add.l d0,d0×5	40
64倍	add.l d0,d0×6	48
128倍	add.l d0,d0×7	56
256倍	add.l d0,d0×8	64

例） 符号付き整数d0の値を1/8倍する

```
	asr.w	#3,d0
```

＊2　MC680x0において操作対象のデータサイズ(オペランドサイズ)がバイトのときとワードのときでは処理速度はまったく同じである。というのも680x0では内部処理的にはすべてワード(2バイト)単位で行っているからである。たとえ操作サイズがバイト指定であっても演算はワードで行っており，処理時間はまったくワードのときと変わらない。結果としてバイトが帰ってくるだけなのである。

＊3　1命令でできる利点はいろいろとある。

たとえばメモリ読み込みにウェイトが入るようなシステム上で，複数の命令を実行すればそのたびに命令の読み込み処理(フェッチ)をMPUが行うわけで，このときにウェイトが入る。ある単一の機能を実現するのに複数の命令を実行しなければならないならば，このフェッチの回数も増えてくるのでそのたびにウェイトが入りすなわち処理が遅くなる。1命令ならばウェイトは1回で済むのに，だ。ウェイト時間がきわめて長いときには複数命令による高速ルーチンよりもちょっと遅めの1命令のほうが高速，なんて場合が出てくるのだ。

また，たとえいまは遅くて複数の命令に分けてコーディングしてあっても将来登場するMPUでは速くなっている場合がある。たとえばここでlslを用いるよりもaddを用いたほうが速いとしているがMC68030ではlsl命令はシフト回数に関係なく同一クロック数で実行できてしまうのでMC68000では，

```
	add.w	d0,d0
	add.w	d0,d0
```

のほうが，

```
	lsl.w	#2,d0
```

よりも速いが，MC68030では逆。2つの命令で実行している前者のほうが遅い。

＊4　MC68030ではなんと32ビットレンジまでに拡張され天下無敵になった。

ジャンプテーブル

掛け算で変換テーブルの話が出てきたが，似たようなものにジャンプテーブルというものがある。これは入力されたある値をキーにして，これに対応したルーチンへジャンプするというものだ。

d0.wに英大文字'A'～'Z'が入力されてくるものとして，これらのアルファベットに対応した26個のルーチンが用意されていたとする。この入力であるd0.wを判別してそれぞれ適切なルーチンへ飛ばしたい。さてどうするか。

なにも考えない場合ならば，

```
	cmpi.w	#'A',d0
	beq	routine_A
	cmpi.w	#'B',d0
	beq	routine_B
	cmpi.w	#'C',d0
	beq	routine_C
	  :
	  :
	cmpi.w	#'Z',d0
	beq	routine_Z
	bra	error
```

のようになるだろう。

これではd0.w＝'Z'のときには'A'～'Y'までの判別処理をパスしてこなければならないことになり実際にルーチンへ飛ぶまでの無駄が多い。そこでジャンプテーブルという概念が登場する。

ジャンプテーブルとは原理的には掛け算テーブルの変換テーブルそのままである。ただし得たい値は演算結果でなく飛び先に関係した値だ。これがジャンプテーブルといわれるゆえんである。

それでは実際にどんなものかを紹介していこう。

いちばん基本的なのは飛び先のルーチンのアドレスを記述しておくものだ。

```
		sub.w	#'A',d0
		add.w	d0,d0
		add.w	d0,d0
		move.l	jump_tbl(pc,d0.w),a0
		jmp	(a0)
	jump_tbl:
		dc.l	routine_A
		dc.l	routine_B
		dc.l	routine_C
		  :
		dc.l	routine_Z
```

リスト先頭でd0.wから'A'のASCIIコードを差し引いている。入力のd0.wは'A'～'Z'なのでこのままではジャンプテーブルを参照するときのキーとしては適当でない。'A'～'Z'はASCIIコードで$41～$5Aだがこれをジャンプテーブルの0番目～25番目の要素と結びつけたいので，

```
	sub.w	#'A',d0
```

を行うことによってd0.w＝0～25に変換している。

ジャンプテーブルにはルーチンの32ビット(4バイト)実行アドレスが格納されているのでこれを取り出すにはd0.w＝0～25のキーを4倍する必要がある。ここで2個並んでいる，

```
	add.w	d0,d0
```

はd0.wを4倍するためのものである。

次に，

```
	move.l	jump_tbl(pc,d0.w),a0
```

だが，これはジャンプテーブルよりルーチンのアドレスをa0.lへ取り出している部分だ。ここまでで，

d0.w＝'A'ならばa0.l＝routine_A
d0.w＝'B'ならばa0.l＝routine_B
　：
d0.w＝'Z'ならばa0.l＝routine_Z

という図式が成り立っている。あとはa0.lが指し示すアドレスへジャンプさせる，

```
	jmp	(a0)
```

で，完了。

ジャンプテーブルを用いると，cmp文の羅列による悪例のときと違い，d0.wがどんな値でも実際にジャンプが起動されるまでの時間が同じという点が優れている。しかし，欠点としてはa0というアドレスレジスタを使ってしまうことが挙げられる。jmp命令のオペランドはアドレスレジスタでなければならないという制約があるのでやむをえずa0にルーチンアドレスを読み込ませている。本来ならば，

```
	move.l	jump_tbl(pc,d0.w),d0
	jmp	(d0)
```

としてd0のみの使用でなんとかしたいところなのだが[*1]。

実はこの欠点を補いつつしかもこれよりも高速なジャンプテーブル実現法があるので紹介しよう。

```
		sub.w	#'A',d0
		add.w	d0,d0
		move.w	jump_tbl(pc,d0.w),d0
		jmp	jump_tbl(pc,d0.w)
	jump_tbl:
		dc.w	routine_A-jump_tbl
		dc.w	routine_B-jump_tbl
		dc.w	routine_C-jump_tbl
		  :
		  :
		dc.w	routine_Z-jump_tbl
```

というものだ。

初めのd0.wに対する加工部分だが，前の方法ではd0.wを4倍していたがこっちの方法では2倍である。これはjump_tblで表されるジャンプテーブル自体が見てのとおりdc.wで構成されているからである。つまりテーブルの1要素がそれぞれワード(2バイト)サイズになったからである。

```
	move.w	jump_tbl(pc,d0.w),d0
```

ではjump_tblから入力d0.wに対応した値をd0.w自身へ読み出している。読み出されたこのd0.wはいったいなにを表しているのか。

ジャンプテーブルは，

　飛び先ルーチンのアドレス
　　－ジャンプテーブルの先頭アドレス

で構成されているが，これは先に紹介した「ディスプレースメント付きアドレスレジスタ間接」の裏ワザ的使用で行った演算に似ている。

そう，この演算はジャンプテーブルの先頭アドレスから飛び先ルーチンのアドレス

までの距離(オフセット)を求めているのだ。すなわちテーブルの各要素はジャンプテーブルをベース(基準)とした飛び先ルーチンまでのオフセットであるわけだ。

次の,

```
jmp     jump_tbl(pc,d0.w)
```

は一見するとなんだかわかりにくいので噛み砕いて説明していこう。

そもそもjmp命令はそのオペランドが指し示すアドレスへ飛ぶ命令である,

```
jmp     LABEL(pc)
```

というアドレッシングを用いることも許されているがこの場合は(pc)が後ろについているが結局LABELというアドレスへジャンプする(当たり前だが)。これがわかるならば,

```
jmp     jump_tbl(pc,d0.w)
```

は,

```
jump_tbl+d0.w
```

へジャンプするということである。いまd0.wには,

```
move.w  jump_tbl(pc,d0.w),d0
```

によってjump_tblから目的ルーチンまでのオフセット値(飛び先アドレス-jump_tbl)が読み込まれているのだからこの式は,

jump_tbl+(飛び先アドレス−jump_tbl)

ということになる。この式はいうまでもなく「飛び先アドレス」ということになる。

で,実際に初めのほうのジャンプテーブル方式と比べてどちらのルーチンが何クロック速いか比べてみよう。

初めのルーチンの総合クロック数は38クロック,後ろのほうは36クロック。わずかながら後者の方法のほうが速い。

速度差はわずかとはいえ,後ろの方法は使用するレジスタがd0だけでほかのレジスタは一切使わないというのが評価できる。また,後ろの方法ではジャンプテーブルがワードサイズなので初めの方法よりもジャンプテーブルのサイズが半分で済む。

しかし,ジャンプテーブルがワードサイズということは飛び先がジャンプテーブルから16ビット範囲内になければならないという欠点も合わせ持つが,まぁよほど大きなプログラムを作成するときでないと問題になることはないだろう*2。

*1 MC68030ではデータレジスタに格納されたアドレスにjmpさせることが可能なので,

```
move.l  jump_tbl(pc,d0.w),d0
jmp     (d0)
```

という表記も可能である。

*2 実際のプログラムでは入力の異常チェックもしなければならないかもしれない。この例の場合でいうならば,入力のd0.wが異常な値('A'~'Z'以外の文字)になっていないかどうかの判断が必要かもしれないということだ。その場合はルーチンの先頭を,

```
sub.w   #'A',d0
bcs     error
cmpi.w  #25,d0
bhi     error
```

とすべきだろう。

小粒で粋な高速テク

ここからはこれまで紹介してきたものよりも小規模なテクニックだが,知っていればやはり得をする。

条件分岐

プログラム中に発生するさまざまな条件に応じて処理を切り換えることのできる条件分岐命令は,応用性の高いプログラムの作成を目指せば目指すほど多用することになる。この部分を高速化するのは実に意味がある。

いちばん基本的な条件分岐は比較命令の直後に使用する場合だ。たとえば,

```
cmpi.w  #10,d0
bhi     bigger
```

ではd0を10と比較して10より大きければbiggerへジャンプ……ということになる。高速テクニックではないが,これをもうちょっと応用した例では,

比較値より大きいとき
比較値と等しいとき
比較値未満のとき

という3つの状態をたったひとつの比較命令の比較結果で条件分岐ができる,というのがある。この例でいけば,

10より大きいとき,
10と等しいとき,
10未満のとき

をひとつの比較命令の結果で分岐させられるというわけだ。

```
cmpi.w  #10,d0
bhi     bigger
beq     equal
*以下,d0<10のとき
☆
```

これはぜひ知っておきたい。条件分岐命令はたとえ分岐が発生しなくても,比較結果である状態(コンディションコードレジスタ:CCR)は保存されるので比較命令の後ろにいくつでも条件分岐命令を記述できるのだ。この例ではd0が10より大きいときにはbiggerへ,そしてd0が10に等しい場合はequalへ,そしていずれでもない場合,すなわちd0が10未満であるときは☆の部分へ処理が移る……ということになる。

ところでMC680x0の持つ比較命令は実はcmp,cmpi,cmpa,cmpm命令だけではない。680x0にはtstという一風変わった命令がある。これはオペランドに与えられたレジスタ,メモリの内容などを見て,内容に応じてCCRを設定するという命令だ。

この命令は主に,オペランドが,

0であるかないか
正か負か

を調べるために用いられる。

よく用いられるのは3ステートスイッチの検査である。オペランドには負値,0,正値の3つのうちいずれが入っており*1,これを判別して,それぞれの値に対応したルーチンへ飛ばすというような使い方だ。

値を調べて0ならば,

```
beq
```

クロックってなんだ?

命令の実行速度の指標として「クロック(実行クロック)」がもっとも一般的ある。普通,このクロック数が少なければ少ないほど処理にかかる時間が少ないということで,つまり「速い」ということである。逆にいうとクロック数の少ない命令を選りすぐって組んだ機械語プログラムは「速い」ということでもある。

ところでこのクロックというのはMPUに供給された特定の周波数を持った発振器のパルスのことであるから,命令の実行クロック数が同じでもMPUの動作周波数が違えばその命令実行時間は違ってくる。10MHzのMC68000よりも16MHzのMC68000のmove命令のほうが高速であるということは誰にでも想像のつくところである。

逆に同じ動作周波数が2種類のMPUに接続されていても,それぞれのMPUがある機能を果たす場合に,実行クロック数が違っていれば,その実行クロック数が少ないほうが速いということである。

このことから違うアーキテクチャのMPU同士を比較する場合その動作クロックや動作周波数の片方の性能数値だけで速度比較するのはまったくナンセンスである。

ここで問題。MPU-Aが動作クロック24MHz,平均命令実行クロック数が4。MPU-Bが動作クロック16MHz,平均命令実行クロックが2だとすると実際高速なのはどちらか。

答え。動作周波数だけ見るとMPU-Aが速いが,実際はMPU-Bのほうが速い。MPU-BはMPU-Aよりも2倍も少ないクロック数で命令を実行してしまうが,MPU-Aの動作周波数はたかだかMPU-Bの1.5倍である。MPU-Aの動作周波数が32MHzならばMPU-Bと同等の速度になるということだ。

で飛ばし，値が負値ならば，

```
    bmi
```

で飛ばす。0または正値ならば，

```
    bpl
```

で飛ばす。この3つの条件分岐を用いて3ステートスイッチの制御を行う。リストにすると，

```
    tst.w   d0
    beq     ZERO
    bmi     MINUS
    *以下0以外の正の値
    ☆
```

こんな感じだ。

d0.wが0ならばZEROへ，d0.wが負値ならばMINUSへ，それ以外すなわち0以外の正値ならば☆の部分へ処理が移る。入力の状態数が多い場合ならば前述したようなジャンプテーブルを用いたほうが速いが，入力がたかだか3状態程度の制御ならばこうしたほうがすっきりしているうえ，それなりに速い。

注意したいのは上で述べたように，

```
    bpl
```

では「0または正値」で分岐してしまうということ。もし，

```
    tst.w   d0
    bpl     PLUS
    bmi     MINUS
    ☆
```

としてしまうと☆の部分には処理が移らない(d0.wが0または正値のときPLUSへ飛んでいってしまい，d0.wが負値のときはMINUSへ飛ぶ。☆部分はいかなる場合も実行されない)。

それでは「0でない正値」であると分岐するような分岐命令はあるのだろうか。また「0または負値」であるときに分岐するような命令があるかというのも気になる。ちょっと調べてみよう。

が，どう調べるか。

条件分岐命令というのはCCRの状態に応じて条件分岐をするのだから，CCRが「0でない正値」「0または負値」を表す状態になっているときに分岐する条件分岐命令があるかどうかを調べればよいのだ。

CCRをじっと見ていてもなにも浮かばないので考えやすいようにcmp命令に置き換えて考えてみる。cmp文の機能定義を見ると，

```
    cmp.w   X,Y
```

は，

```
    Y-X
```

の演算を行い，この結果をCCRに反映するとある。

「0でない正値」は式で表すと，

数値>0

ということになるが，こういう状態になるCCRはこの「数値」の部分をcmp文の比較式「Y-X」に置き換えて，

Y-X>0

としたときのCCRと同じはずである。これは式を変形すれば，

Y>X

となる。Y>Xのとき分岐する条件分岐命令といえば……，

```
    bgt
```

である[*2]。

同様に「0または負値」は，

数値≦0

↓

Y-X≦0

↓

Y≦X

すなわち，

```
    ble
```

だ。

これにより，

```
    tst.w   d0
    bgt     NZERO_PLUS
```

でd0.wが0でなく正の値を持つときにNZERO_PLUSへ飛ばすことができ，

```
    tst.w   d0
    ble     ZERO_MINUS
```

でd0.wが0または負の値を持つときにZERO_MINUSへ飛ばすことができるようになった。

ところで，なにもCCRに値の状態を返す命令はtst命令だけではないことに気づく。MC680x0では加減算命令はもちろん，move命令，ビットシフト命令やあらゆる命令[*3]が，演算結果をCCRへ返すではないか。だからたとえば，

```
    move.b  (a0)+,d0
    ble     end
```

ということもできるのである。これは「(a0)+で読み出したデータが『0または負の値』のときendへ飛ぶ」という制御になる。これを知らないと，

```
    move.b  (a0)+,d0
    beq     end
    bmi     end
```

としてしまうところである[*4]。

```
    lsl.w   #4,d0
    bgt     ビット_exist
```

では，「d0.wを左4ビットシフト(16倍)した結果，d0.wの最上位ビット(符号ビット)が0であり，しかし値が0でないときはビット_existへ飛べ」という感じになる。これを知らないと，

```
    lsl.w   #4,d0
    beq     LABEL
    bpl     ビット_exist
LABEL:
```

のように無駄なラベルを1個作らなければ実現できない処理になってしまう。

比較命令cmpでは，

```
    CMP     X,Y
```

では演算，

```
    Y-X
```

を行った結果によってCCRが変化するといったが，もちろん実際に，

```
    Y-X
```

を行ってもCCRは設定される。すなわち，

```
    sub.l   d0,d1
```

で演算結果d1が負値(minus)または0(zero)になってしまった場合に特定のルーチンへ飛びたいとする場合は，

```
    sub.l   d0,d1
    beq     routine
    bmi     routine
```

とせず，

```
    sub.l   d0,d1
    ble     routine
```

とできる。実際こういう使い方を知らない人が多い。大小結果の条件分岐はどうもcmp命令の後ろでないと使えない……といった覚え込みをしてしまっている場合があるようなのだ。知らなかった人はぜひ覚えておこう。

符号ありと符号なし

ちょっとひとつ基本的なことを注意しておきたい。マシン語では取り扱うデータはプログラマの意思によってその場その場で「符号あり」にしたり「符号なし」にしたりできるのである。C言語のようにどこかであらかじめ「signed(符号あり)」宣言や「unsigned(符号なし)」宣言をしておき，符号ありと符号なしの表記を使い分けなくてはならない，ということはない。

たとえば$FFと$01を比較してもCCRに設定される値(状態)は1通りで，この比較を，

－1($FF)と1との比較

255($FF)と1との比較

とみなすかは条件分岐のときにプログラマが勝手に決められるのである。まぁ「比較」の段階に「符号ありと想定した比較」と「符号なしと想定した比較」の2通りを行って，この両方のパターンの比較結果をCCRに反映している……と考えるとわかりやすい。ちょっと高級言語に慣れてしまっている人には捉えにくいことかもしれない。

＊1　負値には－1，正値には＋1を使用する場合が多い。
＊2　bhiじゃだめかというとだめである。「0でない正値」とある以上，符号なしの比較結果ではだめだ。
＊3　アドレスレジスタに対しての演算はCCRに反映されない。

```
addq.l    #1,a0
suba.l    d0,a0
```

などはアドレスレジスタに対する演算なので計算結果はCCRに反映されず，CCRは命令実行前のまま保存される。
＊4　move命令の実行によるCCR変化はtst命令の変化とまったく同じものなので，レジスタが余っているときなどは，

```
move.b    LABEL(pc),d0
```

のようにレジスタを1個潰して，tst命令の代わりにmove命令を用いたりすることもしばしばである。

フラグを変化させない命令たち

前節でいったようにMC680x0ではほとんどの命令の実行結果がCCRに反映される。intel系やZ80などを使ってきた人たちには奇妙に見えるかもしれないが，これで困ることはない。逆にほかのMPUならば改めて値を試験しなくてはならないものが1命令で済むというような「便利」さを実感する局面のほうが多い。

ところがCCRに反映されない場合もある。これはアドレスレジスタに対して演算を行った場合だ。たとえば，

```
movea.l     LABEL(pc),a1
```

のようなとき，LABELに格納されている値が0であっても，

```
movea.l     LABEL(pc),a1
beq         case_zero
```

のような記述は無意味である。

```
move.l      LABEL(pc),d0
beq         case_zero
move.l      d0,a1
```

のようにする必要がある。d0.lに一度ダミーで値を読み出してCCRを設定させ，そのCCR結果を利用して条件分岐させている。

またアドレスレジスタに対して使用できない命令も結構ある。ビットシフト命令やビット操作命令などだ。アドレスレジスタを2倍したくても，

```
lsl.l       #2,a1
```

のようなことはできない。add命令を使って，

```
add.l       a1,a1
add.l       a1,a1
```

とするしかない。

アドレスレジスタが0かどうかを調べるときにもtst命令が使えないので，

```
tst.l       a1
beq         case_zero
```

というようにはできない。こういう場合には，

```
move.l      a1,d0
beq         case_zero
```

などのようにしてd0へダミーmoveを実行して無理やりCCRに結果を反映させて調べるしかない。

しかしこの特性を逆手に利用すれば高速化の道へとつながる。具体的な例を挙げよう。

マシン語プログラミングにおいて比較的よくYES/NOを返すサブルーチンを必要とする場合がある。たとえばこういう場合。
・a1から指し示されるアドレスにはファイルネームの文字列が格納されており，これに拡張子があるかないかを調べて返す

この場合C言語なんかだとBOOL関数にしてあるならばTRUE，ないならばFALSEかなにかを返すように組むだろう。アセンブラでなら「d0が0ならばない，d0が0以外ならばある」なんて仕様を思いつくかもしれない。この仕様ならば呼び出し側プログラムは(「拡張子有無判定サブルーチン」をcheck_extという名前だとすれば)，

```
bsr         check_ext
tst.l       d0
beq 拡張子なし
以下拡張子あり
```

とこんな感じになるだろう。

拡張子有無判別ルーチンは(仕様：a1から格納されているファイルネーム文字列は終端コード0を持つとする。拡張子の有無はファイルネーム中に文字「.」があるかないかで判断する)，

```
    move.l   a1,-(sp)
loop:
    move.b   (a1)+,d0
    beq      exit_no_ext
    cmpi.b   #'.',d0
    bne      loop
    move.l   (sp)+,a1
    moveq.l  #1,d0
    rts
exit_no_ext:
    move.l   (sp)+,a1
    moveq.l  #0,d0
    rts
```

(リストA)

とこんな感じか。

しかし，たかがYES/NOの結果を持ち帰るのに32ビット長レジスタを1個破壊してしまうのはなんかもったいない。そこで戻り値の仕様を「拡張子があるならばCCRをneに設定する。ないならばCCRをeqに設定する。ただしレジスタはルーチン内で1個も破壊しない」に変えてみる。すると呼び出し側は，

```
bsr   check_ext
beq   拡張子なし
以下拡張子あり
```

となりtst命令がなくて済む。

では，判別ルーチン自体はどうするか。問題は「レジスタを1個も破壊しない(すべて保存する)」だ。

リストAを見ればわかるようにルーチン内でa1とd0は絶対使うからこれらを保存する必要が出てくる。複数レジスタの保存には定番movem命令を使うことになるだろうが，ここでこの命令の仕様をちょっとよく見てみる。そう，movemはレジスタをまとめて読んだり書き込んだりする命令なのだがCCRの内容は更新しない命令なのだ。この習性を利用すると拡張子判別ルーチンは，

```
    movem.l  d0/a1,-(sp)
loop:
    move.b   (a1)+,d0
    beq      exit_no_ext
    cmpi.b   #'.',d0
```

バイブルの間違い？

MC680x0プログラマ必携の本，バイブルともいわれている「68000 PROGRAMMER'S HAND BOOK」(宍倉幸則著 技術評論社)だが，間違いがいくつかあることがわかっている。なかでも条件分岐の動作クロック(同書372ページ)はまったく意味不明なものになっているので2月号で掲載されているOh!X編纂バージョンのものを参照して訂正していただきたい。

またよくいわれるアドレスレジスタ加算命令は，

```
addq.l  #1,a0
```

よりも，

```
addq.w  #1,a0
```

のほうが速いという説，そして相対サブルーチンコール命令は，

```
bsr  LABEL(pc)
```

よりも，

```
jsr  LABEL(pc)
```

のほうが速いという説は明らかに間違いである。MC680x0の動作フェーズを考えた場合明らかに同クロックかかるはず。

しかしモトローラ出の資料の中に同様の間違いがあったという報告例が寄せられており，どうもこのときの誤情報が広く流布されてしまったという見方が強い。68008用のデータが入っているという説もある。

```
        bne         loop
        moveq.l     #1,d0
    exit_no_ext:
        move.l      (sp)+,d0/a1
        rts
```

となる。

ずばり特徴的なのは，最後の，

```
    moveq.l     #1,d0
```

を行ったあとに，

```
    move.l      (sp)+,d0/a1
```

でd0を復元してしまっているところ。これだとd0.lの値はルーチン突入前の値に戻るがmovemはCCRを保存して実行されることから，

```
    moveq.l     #1,d0
```

を実行したときにCCRに設定される「0以外である」は保たれることになる。いうまでもないが，

```
    rts
```

命令もCCRを破壊しないので結局，

```
    moveq.l     #1,d0
```

で設定されたCCRはルーチンを抜けても保存されるのだ。

一方，拡張子がなかった場合は，文字列の終端文字コード「0」を発見するまでループを回り続けることになる。終端文字コード「0」を発見すると，

```
    beq         exit_no_ext
```

の条件分岐が分岐する。このとき終端文字コード「0」を発見したときのCCR「0である」のまま，

```
    move.l      (sp)+,d0/a1
    rts
```

を迎えることになる。

これらの命令群は先ほどもいったようにCCRを保存するので結果としてCCR「0である」のままルーチンを抜けることになる。

初めのルーチンと後ろのルーチンではどちらが速いかは命令の数を数えるだけでも一目瞭然だ。

このようにMPUの制約や特性を味方につければ高速アルゴリズムを導き出せるのだ。

ビット操作で効率2倍

MC680x0系にはbtst，bclr，bset，btchgといったビット操作命令があり，メモリの内容ならば8ビットまで，レジスタの内容なら32ビットまで直値，またはレジスタの値で示したビット内容を操作できる大変高機能な命令だ。

btst.lは「任意のビットが0か1かを検査しCCRに反映する」という高機能ではあるが実に普通仕様な命令だ。

ところがbset,bchg,bclrは実にユニークな仕様である。この3つの命令はそれぞれ順番に「オペランドの任意のビットをON(1に)する」「オペランドの任意のビットを反転する(0⟷1)」「オペランドの任意のビットをOFF(0に)する」という機能を持つが，ユニークなのはそれらの機能を果たす前にそのビットを読み出しCCRに反映するという仕様である。

この，ひとつの命令で2つの機能を果たしてしまうお得な仕様は，MC68000が大型機の高度な割り込み排他処理までもこなせるようにと開発されたかららしい。

さて，このユニークな仕様をなにかに使えないかと考えてみる。

32ビットMPUでビットワークなんてせせこましいことをすると笑われそうだが，MC680x0のビット操作命令は(ビットを使用した)2ステートスイッチ処理をかなり効率よくプログラムできる。

ある処理を一度だけ行い2度目は行わないという処理系を考える。このとき処理を行ったなら行った，行っていないなら行っていないの「覚え」，ワークを設定しなければならない。このままだと話が抽象的になってしまうので，またまた例を挙げるとしよう。

FLAGが0ならばROUTINE_Aを呼ぶ
FLAGが0でないならばROUTINE_Aは呼ばずに次の処理へ進む

のような仕様を考える。FLAGは別に適当なレジスタでもいいがここではメモリ上の1バイト領域とする。普通にプログラムすると，

```
        lea         work(pc),a6
                    ⋮
        tst.b       FLAG−work(a6)
        bne         next_ope
        bsr         ROUTINE_A
        move.b      #1,FLAG−work(a6)
    next_ope:
                    ⋮
    work:
                    ⋮
    FLAG:       dc.b    0
```

のようになる。

「FLAG−work(a6)」云々は前に紹介した「ディスプレースメント付きアドレスレジスタ間接」の裏ワザ的使用テクだ。これの解説はもういいだろう。

まずtst命令でFLAGの内容を調べて，すでにFLAGがON(0以外)ならばROUTINE_Aへのコールは行わないよう条件分岐でスキップさせている。一方，FLAGがOFF(0)ならば，その条件分岐は成立せずROUTINE_Aへのコールへ処理は移る。ROUTINE_Aの処理から帰還したら，「確かにROUTINE_Aは実行したよ」という意味でFLAGをON(ここでは1に)にしている。ここでFLAGがONになったので，もし，再びこの処理系にきても条件分岐，

```
    tst.b FLAG−work(a6)
    bne   next_ope
```

が今度は成立するのでROUTINE_Aへのコールは行われない。

今度は例をビット操作命令を使用したものに改良してみよう。

```
        lea         work(pc),a6
                    ⋮
        bset.b      #0,FLAG−work(a6)
        bne         next_ope
        bsr         ROUTINE_A
    next_ope:
                    ⋮
    work:
                    ⋮
    FLAG:       dc.b    0
```

となる。

一見してわかると思うが，そう，

```
    move.b   #1,FLAG−work(a6)
```

TASマニア物語

「一度行った処理を2度目は行わない処理系」というのを本文で例示しているが，そこで使ったものよりもさらに2クロックばかり速い方法が存在する。それは「TEST&SET」命令「tas」を用いる方法だ。

このtas命令は機能限定版「bset」命令といった感じのもので，これまたひとつの命令で2つの機能という，考えようによっちゃかなり便利なものである。

機能は，

1) オペランドをテストする(tst.b命令に相当)
2) 最上位ビット(第7ビット)をON(1)にする(強制的にON)

となる。

つまり47ページ途中の，

```
    bset.b   #0,FLAG-work(a6)
    bne      next_ope
```

は，

```
    tas      FLAG-work(a6)
    bne      next_ope
```

としたほうが実は速い。

ただし，tas命令ではビットのON動作を最上位ビット（第7ビット）にしか行えないので，ビットワーク制御向きの命令ではないのは確かだ。

まあ，かなりマニアックな命令ではある。

がないのだ。順番にこの例を見ていこう。

まず,

```
    bset.b  #0,FLAG−work(a6)
```

はラベルFLAGで示された1バイト領域のビット0をセットしている。しかし,この命令はこのビットセット操作の前にビットテストを行ってくれる命令であるはずだ。この命令は,

・ビット0が0のときはCCRを「テスト結果は0」に設定してからビット0を1にする。

・ビット0がもともと1のときはCCRを「テスト結果は1」に設定してからビット0を(もともと1だが)1にする。

という動作をすることになる。

つまり,

```
        bset.b  #0,FLAG-work(a6)
        bne     next_ope
        bsr     ROUTINE_A
    next_ope:
```

の部分は,

・FLAGが0ならばFLAGを1にして,

```
    bsr     ROUTINE_A
```

へ進む,

・FLAGがすでに1ならば,

```
    bne     next_ope
```

が成立し,next_ope:へ飛ぶというわけだ。1命令で2の機能を果たしているのをそのまま使っちゃおう作戦,功を奏すというわけだ。実際初めの方法よりも8クロック速い。

こういったビットワークは高級言語熟練者から真顔で「バカらしいから止めなさい」といわれそうだが,場合によってはなかなかうまい方法である。

まず1バイトは8ビットである。そしてビットは0か1の2ステートスイッチであるから1バイトは8個もの2ステートスイッチが実現できる広い領域といえる。

しかし今度は,ビットには「ラベルがつけられないから管理が大変」という指摘がきそうだが,それならラベルもつけてやればいいのだ。

たとえば,

```
    SWORD_FLAG:     equ     0
    ARMOR_FLAG:     equ     1
    SHIELD_FLAG:    equ     2
    HELM_FLAG:      equ     3
    GLOVE_FLAG:     equ     4
    BOOT_FLAG:      equ     5
    RING_FLAG:      equ     6
    ROD_FLAG:       equ     7
```

こんな感じのラベル定義を行って,ワークエリアに,

```
    EQUIPMENT:  dc.b    0
```

なんていう領域があるとすれば,それぞれのスイッチのON/OFFなどは,

```
    bset.b  #SWORD_FLAG,EQUIPMENT−work(a6)
    bclr.b  #SHIELD_FLAG,EQUIPMENT−work(a6)
```

のように行える。各フラグのチェック,反転はいうまでもあるまい*1。こう工夫すればシンボリックに管理できるのでソースの可読性も向上する。

この方式のもうひとつの利点はワークの初期化をまとめて行える点である。たとえば上のFLAG群がもしすべてバイトワークだったならば,

```
    clr.b   SWORD_FLAG−work(a6)
    clr.b   ARMOR_FLAG−work(a6)
    clr.b   SHIELD_FLAG−work(a6)
    clr.b   HELM_FLAG−work(a6)
    clr.b   GLOVE_FLAG−work(a6)
    clr.b   BOOT_FLAG−work(a6)
    clr.b   RING_FLAG−work(a6)
    clr.b   ROD_FLAG−work(a6)
```

のように8バイト分の初期化処理をやらなくてはいけないが*2ビットワークならば8つのスイッチの初期化を1バイトの初期書き込みで済む。

この例ですべてのスイッチを0にするならば,

```
    clr.b   EQUIPMENT−work(a6)
```

となる。

で,ここまできていうのもなんだがMC68000ではテスト命令tstのほうがビット操作命令btstよりも速い。やみくもにビットワークに対してビット操作命令を多用するとバイトワークで管理したほうが高速だったなんてことにもなる。初めの例で挙げたような「1命令なのに2機能でお得」な特性を活かした使い方をしないと高速化とは逆の結果になってしまうことも。ただし初期化処理を頻繁に行う必要があったりメモリの有効利用をしたいというのであればビットワークは効果的だ。

*1 最上位ビットのチェックは,

```
    バイト          btst.b #7,FLAG−work(a6)
    ワード          btst.l #15,d0
    ロングワード    btst.l #31,d0
```

のようにして,

```
    beq
    bne
```

で条件判断/分岐させることができる。しかし,最上位ビットは符号ビットであるから,

```
    バイト          tst.b  FLAG−work(a6)
    ワード          tst.w  d0
    ロングワード    tst.l  d0
```

として,

```
    bpl
    bmi
```

で条件判断/分岐させることもできる。実はこちらのほうがビット検査命令btstを用いた方法よりも速い。

*2 もしそれらがすべて連続した領域に,しかも偶数番地に配置されていたとするならばロングワード命令を使って,

```
    clr.l   SWORD_FLAG−work(a6)
    clr.l   GLOVE_FLAG−work(a6)
```

4バイトずつひとつの命令にまとめて初期化なんてこともできる。

ビットマスクはANDだけじゃない

取り出したデータに対して不要な部分を取り去る処理をマスク処理と呼んだりする。

MC680x0はレジスタは最大32ビット長データまで取り扱えるが,たとえば,

```
    move.b  WORK(pc),d0
```

こうしたとき上位バイトや上位ワードの内容は昔のままである。たとえばもともとd0.1が$12345678というデータでWORKの中身が$55だったとすると,

```
    move.b  WORK(pc),d0
```

によってd0.1=12345655というデータになってしまう。もちろんこういうデータの合成的な使い方もよくするが,このバイトデータを,上位ワードや上位バイトのデータをすべて消し去って32ビットのデータとして使用したいときがよくある。たとえば除算(割り算)命令を使いたいときだ。除算命令は,

続・TASマニア物語

実はtas命令は「Read Modify Writeをサポートしたハードでのみ有効」という,MC68000の命令セットのなかでもかなり特殊な部類の命令だ。

最初,私はX68000がそのような設計になっているのかどうかわからなかったので,こんな怪しそうなものは使わなかいようにしていたのだが(変な機種依存とかすると嫌だし),どうやらX68000シリーズはみんなちゃんとこれに対応しているみたいなのでtas命令は安心して使っていいぞ。

またtas命令は,高速版第7ビットON命令としてももちろん使える。

たとえば,D0レジスタの第7ビットを1にする場合なら,普通に考えていくと,

```
1)  ori.b   #$80,d0     8クロック
2)  bset.l  #7,d0       12クロック
```

なんてのを思いつくだろうが,こういうのはずばり,

```
    tas     d0          4クロック
```

がその目的を達成してくれて,しかも最速である。ちなみに動作クロックを比較すると上の例においてtasは,1)の2倍,2)の3倍も高速である。

32ビット長のデータ

÷16ビット長のデータ

でしか行えない仕様制限を持っている。

よって，この例でWORKの内容を15で割りたいなんてときに，

```
move.b    WORK(pc),d0
divu      #15,d0
```

などとやってしまうと，

$12345655÷15

が計算されてしまう。

こんな場合，

```
move.b    WORK(pc),d0
andi.l    #$000000ff,d0
divu      #15,d0
```

（リストB）

とやってもいいが，MC680x0では符号拡張という専用の命令があるので，これを使うのがよい。

```
move.b    WORK(pc),d0
ext.w     d0
ext.l     d0
divu.w    #15,d0
```

（リストC）

とする*1。

```
ext.w     d0
```

はバイトデータをワード化する命令でいまの例だとこの命令が実行された直後の時点のd0.lは，

$12340055

となる。同様に，

```
ext.l     d0
```

はワードデータをロングワード化する命令である。この例ではこれを実行後には，

$00000055

となる。符号拡張とはそのデータの持つ値を変えずにデータ長を拡張するものだ。たとえばWORKの内容が－1:$ffならば，

実行命令	レジスタの内容
初期値	$12345678
move.b WORK(pc),d0	$123456ff
ext.w d0	$1234ffff
ext.l d0	$ffffffff

のようになる。

ぶっちゃけたことをいうと符号拡張命令extは拡張前の最上位ビットである符号ビットで上位データを書き潰す命令であるということができる。実際，さすが専用の命令だけあってこのextのリストCのほうが初めのandi.lを用いたリストBよりも8クロックも速い。

しかし，たとえばWORKの内容が絶対0～127であるような，つまり正の値であるということが保証される場合ならば，上位バイト/上位ワードは0で埋め尽くされることは当たり前であらかじめわかっていることである。ならば前もってd0を単純に0へ初期化したほうがいいではないか。つまり，

```
moveq.l   #0,d0
move.b    WORK(pc),d0
divu.w    #15,d0
```

としたほうが断然速い。extを2つ用いる手法よりさらに4クロックも速い。ちゃんちゃん。

話的にセコくなるが，この符号拡張を利用してセコいデータ管理系を思いつく。上位バイトにテンポラリ的なそのデータの属性などを設定するワークとして使ってしまい，数値データとして使う場合にはextを用いて一気に消し去り数値データに化けさせる……なんてことができる。Z-MUSICでは楽器のチャンネルID管理にこの手法が使われている。上位バイトに音源の種類下位バイトにチャンネル番号(0-15)を割り当てている。音源の種類は，

$80　MIDI

$00　内蔵FM音源

$01　内蔵AD PCM音源

のように管理していている。

たとえばMIDIのチャンネル3のIDは$8002となる。内蔵FM音源の8チャンネルのIDは$0007となる。このID設計で効率がよいのは，このIDをワードデータの視点で見たときには負値がMIDI，正値だと内蔵音源と判別できるところである。$8002は負値でMIDI，$0007は正値で内蔵音源である。ここでext.wとすると上位バイトは消え去りチャンネル番号を表すワードデータに変身する。チャンネル番号は0-15までの値しか取らないので上位バイトは必ず0クリアされることになるわけだ。

$8002 → ext.w → $0002

$0007 → ext.w → $0007

チャンネル番号のみに変身させられた値は関連ワークなどをインデックス付きアドレスレジスタ間接やインデックス付きPC相対などを用いてアクセスするときには最適の形態となる。

例

```
move.b    ch_work(pc,d0.w)d2
```

こんな最上位ビットをワーク化するID設計はMC680x0系では実に効率がよいので積極的に採用すべきだ。

この「上位バイトをワークにするID」の使用例で「こんなときにこんな命令で目的の処理が実現できる」というのをいくつか示そう。

・最上位ビットを殺(0に)しつつ2倍したい

これはまともにやるならば，

```
andi.w    #$7fff,d0    (ext.w d0)
add.w     d0,d0
```

ということになるだろう。マスクして(最上位ビットを殺して)から2倍して……。しかし，

```
add.w     d0,d0
```

実行時には最上位ビットは絶対外に追い出されることがわかっているので「最上位ビットを殺(0に)しつつ2倍したい」を実現する最良の方法は，

```
add.w     d0,d0
```

だけでいいことになる。

・最上位ビットがONかOFFかを判断しつつ最上位ビットを殺(0に)したい

まともにやるとすると，

```
tst.w   d0
bpl     最上位はOFFだった
andi.w #$7fff,d0  (ext.w d0)
最上位はONだった:
```

とこんな感じだ。d0をtstして調べて負値ならば最上位ビットが立っているということだからこれをマスクする(最上位ビットを殺す)。

さて，ここで先に解説したビット操作命令の「1命令なのに2機能でお得」性を応用すれば，

```
bclr.l    #15,d0
beq       最上位はOFFだった
最上位はONだった:
```

となる。これはbclr命令の使い方そのままと指摘されてしまうかな。

＊1　MC68030では一気にバイトデータからロングワードデータ化する命令extbが装備されている。

マイナー命令「条件付きセット」の活用法

たまにOh!X質問箱などに，

「アセンブラのソースに，

```
st.b    d0
sf.b    LABEL
```

という表記を見かけますがあれはなんなのですか」　（東京都　江楠六八）

という質問がきたりする。

これは「条件付きセット」と呼ばれる命令で確かにインストラクションマニュアルには上のような表記では索引は出ていない。おそらくマニュアルや専門書では「Scc」というような表記で見出しが出され記載されているだろう。

この条件付きセット命令はいわばCISC-MPUならではの高機能命令で，使い方によっては非常に強力なので知らなかった人はこれを機に覚えてしまってほしい。

まず，いくらなんでもbhiやbcsなどの条件分岐は知っていると思う。bはbranch(分岐)の頭文字なのだが，そもそもbの後ろのhiとかcsとはなんなのだろうか。これから見ていこう。

CCRはCMP X,Yのような比較命令等を行ったりすると変化する(設定される)。この比較したX,Yの値の大小関係の組み合わせに応じてさまざまな値がCCRに設定されるわけだが，そのそれぞれに対して下表のような名前がつけられているのである。

●CMP X,Yを行ったあとのCCR

大小関係	符号なし
X<Y	hi(HIgh)
X≦Y	cc(Carry Clear)
X=Y	eq(EQual)
X≠Y	ne(Not Equal)
X>Y	cs(Carry Set)
X≧Y	ls(Lower or Same)

●CMP X,Yを行ったあとのCCR

大小関係	符号あり
X<Y	gt(Greater Than)
X≦Y	ge(Greater or Equal)
X=Y	eq(EQual)
X≠Y	ne(Not Equal)
X>Y	lt(Less Than)
X≧Y	le(Less or Equal)

条件分岐命令ではbの後ろにこれらのCCRの状態名を書いて，

```
	bls	LABEL
```

などのように記述した。むろんこれは比較結果が符号なしでX≧Yのとき分岐せよという命令になる。

で，条件分岐は条件成立で分岐(ジャンプ)するが，このマイナー「条件付きセット」は条件が成立すると指定されたオペランドに$FFを書き込む動作をするのだ。条件不成立だと$00を書き込む。すなわち条件成立時には，

```
	move.b	#$ff,???
```

に，条件不成立時は，

```
	clr.b	???
```

に化けるということもできる。

応用の仕方はいろいろだが基本的には比較結果をどこかのフラグワークに覚えておく，なんていう目的で使う。

たとえば，

・d0.wが100以上ならばFLAGを$FFへ，99以下ならばFLAGを$00に設定する。

という場合を仮定すると，

```
	cmpi.w	#100,d0
	sge	FLAG-work(a6)	*1
```

という感じになる。また条件分岐を組み合わせることも可能で，

・d0.wが100以上ならばFLAGを$FFへ，99以下ならばFLAGを$00に設定する。

・d0.wが符号なし整数で101以上ならばルーチン名over_101へ飛び，ちょうど100ならばルーチン名equal_100へ飛ぶ。

という仕様を想定したとすると，

```
	cmpi.w	#100,d0
	sge	FLAG−work(a6)
	bhi	over_101
	beq	equal_100
```

とこんなことができる。

条件付き命令「Scc」は実行後もCCRを保存しているのでその他のCCR条件判別命令(たとえばこの例のような条件分岐)に繋げることができるのだ。

ところで実はこの条件付き命令には裏ワザ的使用方法がある。それは$00や$FFをオペランドへ書き込む命令として使用する場合だ。

```
	move.b	#$FF,???
```

や，

```
	move.b	#$00,???
```

と比べてこの条件付きセット命令は高速，しかも命令語長も短いという特徴があるのだ。

しかし「『条件』付きセット命令なんだからCCRの状態に応じて，

```
	move.b	#$FF,???
```

にも，

```
	move.b	#$00,???
```

にもなってしまうのではないか」といわれそうだ。しかし条件付きセットにはBccにはない，

状態	条件名称
常に条件成立	T(always True)
常に条件不成立	F(always False)

という条件名称があり，この2つの条件名称を使ったScc命令はCCRの状態を無視することができるのだ。

```
	sf	???
```

ではCCRの状態によらず常に条件不成立になり，オペランドに$00を書き込むことができる。また，

```
	st	???
```

ではCCRの状態によらず常に条件成立になり，オペランドに$FFを書き込むことができる。

そして，

```
	sf	d0
```

は，

```
	move.b	#$00,d0
```

よりも4クロック速く，

```
	st	d0
```

も同様に，

```
	move.b	#$FF,d0
```

よりも4クロック速い。オペランドをデータレジスタにした以外のときではmove.b表記と等速だが，マシン語コード長が同機能を実現するためのmove.b表記よりも短くなるという長所を持つ。この比較をまとめたものを表3に示す。

*1 もういい加減わかってもらえてると思うがすでに解説したディスプレースメント付きアドレスレジスタ間接の裏ワザ的使用の例である。

ループ制御命令でGO!

MC680x0にはこれまたCISCならではの高機能な命令，ループ制御命令なるものが装備されている。Z80はDJNZというBレジスタをループ変数に使えるループ制御命令を持っていたが，これよりも汎用性のある命令をMC680x0は持っている。

基本的なところから解説をしていこう。

たとえばループ変数をd0にして10回ROUTINE_Aをコールしたい場合ならば，

```
	moveq.l	#10−1,d0
loop:
	bsr	ROUTINE_A
	dbra	d0,loop
```

表3 無条件セット命令

表記例	move.b #$00,??と比べて
sf d0	move.b #$00,d0よりも 4クロック速く マシン語コード長も短い
sf (a0) sf (a0)+ sf LABEL(a0) sf LABEL(a0),d0.w sf LABEL.W sf LABEL.L	実行速度は move.b #$00,?? と同等だが マシン語コード長は これより短い
sf −(a0)	実行速度は move.b #$00,−(a0) よりも遅いが マシン語コード長は これより短い

表記例	move.b #$ff,??と比べて
st d0	move.b #$ff,d0よりも 4クロック速く マシン語コード長も短い
st (a0) st (a0)+ st LABEL(a0) st LABEL(a0),d0.w st LABEL.W st LABEL.L	実行速度は move.b #$ff,?? と同等だが マシン語コード長は これより短い
st −(a0)	実行速度は move.b #$ff,−(a0) よりも遅いが マシン語コード長は これより短い

のようになる。

注意すべきなのはループ変数に,

ループしたい回数−1

の値を設定しなければならない点だ。Z80からやってきたマシン語フリークには「??」と思えるかもしれないが，このループ変数をインデックスとしてメモリの内容を参照したりするときは，こちらのほうが便利である。たとえばWORKで指し示される内容を次々に参照していきたい場合,

```
        lea         WORK(pc),a0
        moveq.l     #10-1,d0
loop:
        move.b      (a0,d0.w),d1   (*1)
          ⋮
        dbra        d0,loop
```

とすることにより,

ループ1回目(WORK+9)の内容を参照
ループ2回目(WORK+8)の内容を参照
ループ3回目(WORK+7)の内容を参照
⋮
ループ9回目(WORK+1)の内容を参照
ループ10回目(WORK+0)の内容を参照

とWORKから始まる領域の先頭から10バイトを参照できる。配列でいうとWORK[0]〜WORK[9]を参照できるというわけだ。

もしループ変数が10〜1まで変化するZ80,DJNZパターンだとすると,

ループ1回目(WORK+10)の内容を参照
ループ2回目(WORK+9)の内容を参照
ループ3回目(WORK+8)の内容を参照
⋮
ループ9回目(WORK+2)の内容を参照
ループ10回目(WORK+1)の内容を参照

となりWORK+0すなわち配列の要素の1番目WORK[0]は参照されないことになる。

まぁどちらがいいかという議論は，おいておいて，とにかくMC680x0のループ制御命令は「希望ループ回数−1」をループ変数に設定するんだなと暗記しておこう。

ところでこのループ制御命令はインストラクションマニュアルには,

DBcc

という見出しで載っているはずである。「DBcc」ということは49ページで紹介したCCRの状態名が書けるということである。あまりdbraの書式以外は用いないかもしれないが，一応,

```
        dbmi
        dbeq
```

といった表記ができるということである。これはいったいどんな機能をもたらしてくれるのか。

結論からいうと，これはループ終了条件をもうひとつ追加するものである。

たとえば,

```
        dbmi        d0,loop
```

ならば「(d0をループ変数として)指定回数ループを完了したか，あるいはCCRがmi(minus)のときループを終了する」ということになる。ちなみにdbraは実はdbf,つまりccがf(always false)のDBcc命令である。fということは,

常に条件不成立
↓
CCRの状態に関係なくループを実行
↓
ループの終了条件は指定回数ループを終了したときのみ

となる。

dbfではなんかわかりにくいのでdbraという表記を許しているのである。

dbfがあればもちろんdbtもあるわけで，こちらは,

常に条件成立
↓
CCRの状態によらずループを終了

となり，単にループ変数を1減らす命令になってしまう。はっきりいって使われることはほとんどない。

さて，ループ終了条件をひとつ増やせるのがわかったところで，いったいどういう使い道があるのかわからないという人もいるだろう。便利と感じることのできる例を考えてみよう。

a0から指し示される領域に文字列があったとする。この文字列はコード0でその文字列の終端を表現していたとする。すなわち，マシン語プログラムソースのデータ領域に見られる,

```
        dc.b 'HELLO!',0
        dc.b 'MY NAME IS GERRY VANE.',0
```

のパターンが存在しているということだ。

さて，このとき，このa0から格納されている文字列を，a1で示される別の領域に転送したい。

しかしa1の領域は20バイトしか確保されておらず，20バイトに満たない文字列ならばすべてa1に転送したいが，20バイトを超えている場合は20バイト転送した時点で，転送を打ち切りたい。

この条件で普通にプログラムすれば,

```
        moveq.l  #20-1,d0
loop:
        move.b      (a0)+,(a1)+
        beq         exit_loop
        dbra        d0,loop
exit_loop
```

とするだろう。しかしよく見るとこのプログラムは,

```
        beq         exit_loop
```

でループを抜け出しているので「CCRがeqになったらループを抜ける」というループ脱出条件をdbra(dbf)に付け加えればよいことになり，すなわちこれはdbeqにまとめることができるではないか。

つまり上記リストは,

```
        moveq.l  #20-1,d0
loop:
        move.b      (a0)+,(a1)+
        dbeq        d0,loop
```

のようにできるということである。a0の内容が,

```
        dc.b 'HELLO!',0
        dc.b 'MY NAME IS GERRY VANE.',0
```

だとすると転送結果は,

```
        dc.b 'HELLO!',0
        dc.b 'MY NAME IS GERRY VAN'
```

となる('HELLO!',0は20文字未満なのですべて転送されるが'MY NAME IS GERRY VANE.',0は終端コード0も含めると23文字

*.Xファイルの謎 #1

Human68kで実行できるプログラムファイルには「*.R」「*.Z」「*.X」の3種類だ。

「.R」ファイルはメモリ上のどのアドレスに読み込まれたとしても動作するプログラム，いわゆるリロケータブル(Relocatable/再配置可能な)プログラムである。16Mバイト(あるいは4Gバイト)空間のどのアドレス上でも実行可能だ。そして反対に「.Z」は実行アドレス固定のプログラムファイルで，そのプログラム固有の絶対アドレスにロードされなければ実行できないプログラムである。馴染みの深い拡張子「.X」のファイルは……というとユーザーからみればリロケータブルなプログラムであるが，実は少々特殊なファイルなのである。

*.Xファイルは確かに実行アドレスを制限していないが，相対アドレッシングだけを使ったプログラムかというとそうでもない。平気で,

```
        lea         LABEL,a0
        move.b      #15,WORK
        jmp         LABEL
```

のような32ビット絶対アドレッシングをバリバリに使ったプログラムをアセンブルして.Xファイルを作成することはしょっちゅうだ。

実は*.Xファイルの先端と終端にはリロケート情報というものが付属しており，プログラムがロードされると，この情報をもとに，プログラム中で使用された絶対アドレッシング部分を適切なアドレス値に置き換えるのだ。これはプログラム実行前に行われる。もちろんやっているのはOSであるHuman68kサマだ。

なので先頭から20文字までしか転送されない)。

またすでに紹介したようなYES/NOを返答してくれるサブルーチンで，たとえばサブルーチンを10回実行したいが実行結果がmi(minus)ではサブルーチンがエラーを起こしているという意味なのでループを抜けたい……という場合も，

```
    moveq.l  #10−1,d0
loop:
    bsr      subroutine
    dbmi     d0,loop
```

とできる。

＊1　実はここを，
move.b(a0)+,d0
ようにしてしまえば，仮にMC680x0のループ制御命令のループ変数が10〜1と変化するZ80，DJNZパターンであったとしてもWORK[0]〜WORK[9]までを参照できる。しかしa0はインクリメントされて破壊されてしまい再び同様の処理を行うときにはこのa0の内容をワーク先頭に再設定しなければならない手間が発生する。

データを上へ下へ

MC68000は設計上の制約から奇数アドレスからワード(2バイト)データやロングワード(4バイト)データを読み出すことができない(MC68030では可能)。行った場合はアドレスエラーが発生する。とはいえ，奇数アドレスからワードやロングワードデータを拾ってきたいということはよくあることだ。

ワードデータを奇数アドレスから読み出す場合は，

```
    move.b  (a0)+,d0
    lsl.w   #8,d0
    move.b  (a0)+,d0
```

となるだろう。シフト命令で最初に呼んだバイト値を上位バイトへスライドさせて，空いたところに次の下位バイトとなる値を読み込む。

ロングワードならば，同様なアルゴリズムを用いれば，

```
    move.b  (a0)+,d0
    lsl.l   #8,d0
    move.b  (a0)+,d0
    lsl.l   #8,d0
    move.b  (a0)+,d0
    lsl.l   #8,d0
    move.b  (a0)+,d0
```

となるだろう。

ちょっと賢い人ならば最初のlslは絶対上位ワード(ビット16〜31)まで有効な値がシフトされないから，

```
    move.b  (a0)+,d0
    lsl.w   #8,d0
    move.b  (a0)+,d0
    lsl.l   #8,d0
    move.b  (a0)+,d0
    lsl.l   #8,d0
    move.b  (a0)+,d0
```

とできる，と工夫することだろう。これはもっと工夫できる。最初のワードデータを読み込んだ時点でそのワードデータをまるごと上位ワードへシフトしてしまえば後半のワードデータ読み込み時のシフトは，

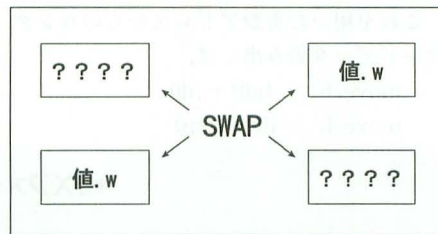
```
    lsl.w   #8,d0
```

でできるので高速化できると考えられる。しかしワードデータをまるごと上位ワードへシフトしたくてもlsl命令のシフト回数値は8までなので，16回シフトするには，

```
    lsl.l   #8,d0
    lsl.l   #8,d0
```

としなければならず，

```
    move.b  (a0)+,d0
    lsl.w   #8,d0
    move.b  (a0)+,d0
    lsl.l   #8,d0
    lsl.l   #8,d0
    move.b  (a0)+,d0
    lsl.w   #8,d0
    move.b  (a0)+,d0
```

これではかえって遅くなってしまう。ところがMC680x0では上位ワードと下位ワードを入れ換えるswap命令というのがあり，これを使えば下位ワードの値を上位ワードへ1命令(しかもたった4クロックの実行速度で速い)で持っていくことができる。しかしもともと上位ワードにあった正体不明の値が下位ワードへ降りてきてしまう。

が，これは下位ワード読み込み，

```
    move.b  (a0)+,d0
    lsl.w   #8,d0
    move.b  (a0)+,d0
```

で書き潰すことができる。結局，

```
    move.b  (a0)+,d0
    lsl.w   #8,d0
    move.b  (a0)+,d0
    swap    d0
    move.b  (a0)+,d0
    lsl.w   #8,d0
    move.b  (a0)+,d0
```

とでき，初めのlslだけを用いた例(104クロック)よりも24クロックも速い。

このようにswap命令を場合によっては，

```
    lsl.l   #16,d0
```

のように使用することもできるのだ。もちろん上位ワードを下位ワードへ持ってくるという目的で使用するならば，

```
    lsr.l   #16,d0
```

のようにも使えるということである。これは覚えておきたい。

もちろん下位データを上位データへ，上位データを下位データへという処理はワード単位ではなくバイト単位で行いたいときもある。ワード単位のときはこのswap命令を用いれば一発で実現できるが，たとえば下位バイトを上位バイトへとか上位バイトを下位バイトへのような処理はどうしたらよいのか。

素直に考えれば，

・下位バイトを上位バイトへ

```
    lsl.w   #8,d0
```

・上位バイトを下位バイトへ

```
    lsr.w   #8,d0
```

とすることができる。これらの処理はMC68000ではともに22クロックかかる命令である。ところがこれよりも6クロック速い方法が，なんと2つの命令を組み合わせて実現できるのである。

まずその方法から紹介しよう。

・下位バイトを上位バイトへ

```
    move.b  d0,−(sp)
    move.w  (sp)+,d0
```

・上位バイトを下位バイトへ

```
    move.w  d0,−(sp)
    move.b  (sp)+,d0
```

なんとスタックを使った裏ワザである。Z80的テクニックというべきか。確かに8クロック命令を2つ使用しているので実行速度はともに16クロック。

この裏ワザの仕組みを順番に解説していこう。

まず，このテクニックの特徴的な点はスタックを利用しているということと，プッシュ時とポップ時のデータサイズが異なるというところである。スタックを利用するのはともかく，プッシュ時に，

```
    move.b  d0,−(sp)
```

でスタックを−1して，

```
    move.w  (sp)+,d0
```

でスタックを+2したらスタックの整合性が失われてマズいんじゃないのーと，突っ込みがきそうである。ところがスタックに対しての−(sp)や(sp)+はたとえ命令サイズがバイトであってもワード単位で処理が行われるのである。これはもちろんMC

68000がワードデータやロングワードデータを奇数番地に対してアクセスできないという制約からくるものだ。

たとえば,

```
move.b    d0,-(sp)
move.w    d0,-(sp)
```

という命令列の場合,もともと偶数アドレスだったスタックが,もし1行目のバイトデータの-(sp)によってスタックを-1してしまったらスタックは奇数アドレスになってしまい2行目のワードデータの-(sp)は奇数に対して行うことになってしまう。これは明らかにMC68000の制約に違反している。よってこういうことにならないよう,バイトデータのプッシュもスタックを(ワードデータのプッシュと同じように)-2し,それからバイトデータを書き込む仕組みになっている*1。

具体的な例を示そう。

↓スタック

□□ □□ □□ □□ □□ □□ □□

スタックがこの位置とする。そして説明をわかりやすくするために□は1バイト領域を表し,□□はワード領域を表すとする。

いま,

```
move.b    d0,-(sp)
```

とすると,-2してからそのバイト値を書くから,

のように■の位置にバイトデータが書き込まれるはずである。あくまで-2してからのバイトデータの書き込みなので,■の右となりの□についてはまったく触られない。

さてこの状態で

```
move.w    (sp)+,d0
```

とすれば,

d0には■□が入ってくるはずである。これはつまり,

```
move.b    d0,-(sp)
```

でプッシュされたバイトデータが上位にきていることにほかならない。ただし,□の部分はまったく無関係な値が格納されていることに注意。つまり,

```
lsl.w     #8,d0
```

では下位バイトが上位バイトに移動すると同時に下位バイトは0になっているが,この手法だとスタックの領域に前からあった値が下位バイトに乗ってきてしまうということだ。具体的にいえば,たとえばいま,d0.w=$ABCDだとすると,

```
lsl.w     #8,d0
```

ではd0.w=$CD00になるが,

```
move.b    d0,-(sp)
move.w    (sp)+,d0
```

ではd0.w=$CD??となってしまうということである(??の値はなんだか予測できない)。ただ,下位バイトにはなにかすぐ別の値を入れたりする場合ならばmove.bで上書きできるし,さらに0にしたい場合でも,

```
move.b    d0,-(sp)
move.w    (sp)+,d0
clr.b     d0
```

とすればよい。これでも,

```
lsl.w     #8,d0
```

より2クロック速い。

さて,逆動作である,

```
move.w    d0,-(sp)
move.b    (sp)+,d0
```

もやはり完全に,

```
lsr.w     #8,d0
```

とは互換ではない。たとえばd0.w=$ABCDで,

```
lsr.w     #8,d0
```

ではd0.w=$00ABとなるが,

```
move.w    d0,-(sp)
move.b    (sp)+,d0
```

ではd0.w=$ABABと上位バイトの内容が下位バイトにコピーされるような動作が行われる。この動作の仕組みの解説は省略するが,各自で先ほどのように1ステップずつ考えてみると,あぁ,なるほどと思えるはずである。

これを用いた奇数アドレスからのロングワードデータ読み出しは,

```
move.b    (a0)+,d0
move.b    d0,-(sp)
move.w    (sp)+,d0
move.b    (a0)+,d0
swap      d0
move.b    (a0)+,d0
move.b    d0,-(sp)
move.w    (sp)+,d0
move.b    (a0)+,d0
```

とすることができ,全部lsl.lを用いた例よりも36クロックも速いことになる。とはいえ,上のリスト,一見しただけじゃなにをやってるかわからないのが欠点かも……。

*1 バイトデータのポップは逆に値を読んでからスタックを+2している。

終わりに

思いつくままに書いてきてしまった。内容も基本ワザから変態マニアックワザまでがゴチャ混ぜになってしまった気もする。MC68030の基本テクなども紹介したかったが,誌面の都合でそれらはまたいずれ。

ところで,今回紹介したテクニック以外にもさまざまなテクニックが開発されていると思う。俺はこんなの知ってる,使ってる! というのがあれば読者ハガキでもなんでもいいからOh!X編集部まで送ってほしい。ユニークなものはどんどん紹介していきたいと考えている。

・参考文献

「68000 PROGRAMMER'S HAND BOOK」,宍倉幸則著,技術評論社

「アセンブラマニュアル」(C Compiler ver.2.1に同梱),シャープ

*.Xファイルの謎 #2

*.Xファイルに付属するリロケート情報は基本的にプログラム中で使用された絶対アドレッシングに比例して増えてしまう。本文にもあるように相対のほうが速度的にも速く,32ビットの絶対アドレッシングは使わないにこしたことはないということである。たとえば,

```
        lea       $e90001,a0
```

のように,I/Oアドレスのような直値指定の場合はリロケート情報は発生しない。どこで実行されようと,I/Oアドレスは不変だからだ。

```
LABEL:
        lea       LABEL,a0
```

はLABELのアドレス値はプログラムの実行アドレスが変われば変化するのでリロケート情報が発生する。もちろんLABEL(pc)にすれば相対指定になるので発生しなくなる。

また,ジャンプテーブルに関しても本文で示した,

```
jump_table:
        dc.l      ROUTINE_A
        dc.l      ROUTINE_B
```

のようにdc擬似命令で32ビット実行アドレスを並べたりするものはリロケート情報が発生し*.Xファイルは膨れあがる。理由は上と同じだ。

```
jump_table:
        dc.w      ROUTINE_A-jump_table
        dc.w      ROUTINE_B-jump_table
```

のタイプは完全な相対値をdc疑似命令で書き込んでいるだけなので発生しない。

まぁ,プログラム中のラベルを相対指定でなしに用いてしまうとリロケート情報が膨らむ,くらいに覚えておこう。

コーディングの深みにはまる

コスい技を磨く

Yokouchi Takeshi 横内 威至

無駄な処理を省き，資源を使いきることが高速化の道である
レジスタを最大限に生かすために，パズルのようにルーチンを組み上げる
コーディングの芸術を目指して，より実戦的なテクニックを見てみよう

コスい技特集，というとやっぱりカゲ。間合を敵からちょっとおいて，前後にピストン運動を繰り返し，突然近づいて投げる。時間がなくなってヤバイときは飛んでいけば相手もびっくり。立つかしゃがむかの早押しゲームが楽しめる。

死ね，クソッタレ。そんなのはVF2の正しい楽しみ方ではないのだ。PKもやめろ。もっと美しい倒し方を知らないのか。いかに敵を読むか，いかに敵を欺くか，がVF2の勝負方法だ。まあ，頭の弱い連中のことだ。美的センスに欠けるのは仕方ないことだろう。勝ちだけに執着する，ミジメな，クソよりもくだらないヘボいプレイは弱者の証，下衆な心の乏しい浅はかなあなたにピッタリ。

＊　＊　＊

あ，関係ないっすね。ということでコスい技特集，といえばマシン語。なぜかマシン語のコスい技は美しい。究極の解法とでもいうのか，極限の効率を追求する凄まじいコード。当然，表向きには理解しにくい形となっていることも多い。単純な目ではその真髄を知ることは許されない。最先端の芸術とは常に理解し難い面をもっているものだ。ダリが糞尿にこだわったように，ピカソが表面的な型，原形にとらわれずに物事を見ていたように。

ちょっと高度そうなことを書くとボロがでるからこの程度にして，わけがわからないがとりあえず本質を理解するのは厳しいという点でも，これらの難解なコードたちは芸術だ。

個々における芸術があなたに及ぼすものは，たいていは心地よい共感かもしれない。しかし，新鮮に感じるものにはショッキングな感動がともなうものだ。激しい嫌悪感かもしれないし，とりあえず得体の知れない力に心が踊るものである。その非日常的な感覚こそが芸術を味わう意味となる。理解しようとしまいと，芸術にはそのような力がある。

この得体の知れないコードを解析し，理解するのは平凡な努力ではなしえない。芸術の本質を見極めるのは常に正しい，理論的な分析が必要なのである。

それでは，巷にころがるこれらの芸術の片鱗を探ってみようではないか。

初めに

初めにいっておくが，ここでは基本的なテクニックというのは無視しておきたい。というのは，たとえばアドレスレジスタはadda.wよりもlea.lを使おう，とかは命令表を見ればわかることだから，こと細かく追及しないことにする。アドレッシングなどを使った技については西川氏の記事を参照していただきたい。

また，実は68000系CPUはマシン語の基本性能が優秀だから，Z80のようにさまざまなテクニックのようなものがあるとは思えないので，どちらかというと一般的に使う機会の多いような処理のアルゴリズムを書いてみようと思っている。また，質は問わないことにする。アルゴリズムの質を問うならば，過去の「X68000マシン語プログラミング」を読み返すこと。さすがにレベルの高い内容だ。

では具体的に進めよう。各項目の★は難易度，技のキレなどの総合評価。最低は1つ。最高は決めてないので気分次第。アキラの右端脚→揚抱だと★4つぐらいだと思えばちょうどいい。

演算関係

●小数点演算　★

結構基本ではあるが，演算の基本となるのであらゆる方向に応用がきく。16ビットのうち，上位ビットを整数部，下位ビットを小数部として考える。

たとえば1.5+2.75は，

```
move.l   #$18000,d0
move.l   #$2c000,d1
add.w    d1,d0
swap.w   d0
```

とすればd0.wに整数部分の4，上位ワードには4.25の小数部分を意味する$4000が入っている，というようにきわめてくだらない内容だ。説明する暇があったらアキラ5段を練習しているほうがよかった。

ではなんに応用するかである。まあ好きなように応用すればよし。目立つところではラインルーチンなんかが美味しい。普通はDDAだったかDHAだったかは忘れたが，そんなようなアルゴリズムが一般的だけど，処理速度の安定感から私は小数点で処理するほうが好みだ。

●10進数，2進数変換　★

たまに使うかもしれないから解説を。これも簡単な処理だが，たとえば16ビットの数値だったら10000×A＋1000×B＋100×C＋10×D＋Eというように表現するだけで解決。順番に割っていけばA～Eにあたる数字が出てくる。abcd.bとかの命令は応用がきかないのでできれば使いたくない。これまたくだらない内容で申しわけない。

先ほどの小数点処理と絡めればd0はd1の何倍か，なんてのは簡単に表示できる。

●32ビット÷16ビット　★★

ただのdivs.wでは制限がある。当然オーバーフロー，結果が16ビットで表現できない計算はエラーとなるという制限のことだ。これのせいで，divs.wはかなり気を使う命令となってしまっている。

d0（32ビット）÷d1（16ビット）をd0に32ビットで答えるには以下のとおり。ただし全部正数とする。

```
move.w   d0,d2
clr.w    d0
swap.w   d0
divu.w   d1,d0
```

```
    swap.w   d0
    swap.w   d2
    move.w   d0,d2
    swap.w   d2
    divu.w   d1,d2
    move.w   d2,d0
```

まず(d0÷65536)÷d1を求め，余り(＝Aとすると(65536×A÷d1)<65536) の65536倍と，最初のd0の下位ワードの合計をd1で割れば結果が出ることになる。よく考えれば理解できると思う。ちょっと嬉しいテクニックかな。

また，同じようだが小数点まで求めるのは次のようにやる。ただし，結果が65536以下になることに限定しなくてはならない。

```
    divu.w   d1,d0
    move.w   d0,d2
    swap.w   d2
    clr.w    d0
    divu.w   d1,d0
    move.w   d0,d2
```

結果はd2に入る。上位ワードが整数部，下位ワードが小数部。先ほどの小数点演算とのコンビネーション風だ。

●**16ビット÷16ビット　★★★★**

これは読者の方，土井宏治さんの投稿によるアルゴリズムだ。16ビットの割り算というものに目をやっていなかったため，私はかなり驚いた。

これはA÷Bを，最初の小数演算の応用のように，

A×(1÷B)

と置き換えるというものである（逆数の掛け算)。

当然，1は65536，実際はオーバーフローの関係で16384で表すことになる。a0が逆数テーブルを指していて，d0＝A，d1＝Bとすると，

```
    ext.l    d1
    add.l    d1,d1
    muls.w   (a0,d1.l),d0
    add.l    d0,d0
    add.l    d0,d0
    swap.w   d0
```

となる。なんと，これだけで精度はやや落ちるが除算ができてしまう。

テーブルの内容は，

```
            16384÷－32768
            16384÷－32767
                 ⋮
                 ⋮
テーブル：32767
            16384÷1
            16384÷2
                 ⋮
            16384÷32767
```

となっており，'テーブル'の部分をリストでのa0が指していればよい。ここの32767は＋無限大を意味する。

このアルゴリズムの導入による実行速度アップは約1.5倍ぐらいだろうか。単発の命令を変化させるものとしては究極の形であろう。

これはかなり使える。精度はやや下がる(±1ぐらいズレる)が，用途によってはまったく問題ない。テーブルの値を適当にいじったりすればさらに応用できるし，実際SLASHにも応用させていただいた。ありがとう。

＊　　　＊　　　＊

以上が演算に関してである。小数演算が基本となるのはわかってもらえただろう。そして最後の除算，これもかなり応用できる。この2つがあるだけでかなりいろいろなことが高速化できるに違いない。除算自体，制限がいろいろとある。

しかし，使用する数値のとりうる範囲をよく調べればエラーを処理する必要がなかったり，mulu.wで十分であったりということも多い。数値演算での高速化のカギは数値の取りうる範囲だ。これに着眼すれば不必要な手間さえも省くことができる。頑張れ。

休憩

ここでマシン語でプログラムを書くときの定石を述べておこう。定石，といっても私のは我流だから一般的ではないかもしれない。

まず，ここで挙げたような基本処理はマクロにしてしまう。最初，私はマクロをまったく知らなかった。Z80の頃にはそんなものは使えなかったため，私はそのままでしばらくやっていた。プログラムのレベルも上がると，次第に大きいリストが多くなる。しかし，それは無意味に長い部分が多くなっていたにすぎなかったのだ。視認性の問題からも，ちょっとしたまとまった処理はマクロを使うこと。また，これによってサブルーチンを使う機会が減る。実行ファイルは膨大になりがちだが，メモリはいくらでもある。処理速度のほうが大事。サブルーチンは極力避けるべし。

また，無意味な命令，すなわちLINKなんかはアセンブラでは無意味。スタックを通して変数を渡すのも避ける。どうせレジスタは16個もあるのだ。有効に使うべきである。ちなみに，最近ではレジスタが足りなすぎて困っている。やはりRISCの時代なのか？

足りないレジスタは頑張って処理順序を考えて可能な限り効率化させる。これを行えるのがマシン語の最大の価値である。C言語の吐き出すコードなんかはクソ以下。参考にすべき部分はまったくない。別にCを非難しているわけではない。アセンブラの有効性をもっと認めるべきだといいたいだけだ。いつの時代でもマシンの本当のスペックを引き出すのはマシン語しかないのだ，多分。

画像関係

●**アドレス計算　★**

基本的なアドレス計算。でも，1ラインずれると$400も違って困る。最高で$80000もあるから非常に嫌だ，と思うけどたいしたことはない。テーブルにするのも楽だし，ビット操作で楽勝。d0.wがX座標，d1.wがY座標としてa0レジスタにアドレスを求めるには，

```
    swap.w   d0
    clr.w    d0
    ror.l    #6,d0
    add.w    d1,d1
    add.w    d1,d0
    movea.l  d0,d1
    adda.l   #$c00000,a0
```

となる。最初の3行はd0を$400倍している。ややシブいのは最後の2行。本来ならばlea.lのあとd0を加えたいが，このほうが2クロック節約。まあこれだけ。

●**描画自体について　★**

基本的にはブロック転送だとかブロック書き込みなどのことだが，X68000では画面モードなんかも考慮しなければならないことになる。たとえば65536色モードでは1ドットあたり1ワードだが，256色モードでは1バイトであり，連続するドット間では1バイトが空領域となる。

まず，クリアなどの同一データをダラダラと書き込むのはmove.wを羅列するのが楽。書き込む長さをキーとして適当にこの部分にジャンプさせれば自然とループ展開になるのは基本。当然，move.lのほうが高速だ。また，さらに派手に展開を行ってmovem.lを使うのがもっとも速いだろう。ただし，指定した長さだけ書き込むためのルーチンを展開するのはきわめて面倒だ。

特定のデータを小さい領域にコピーする

だとか描画するとかならばmovepも有効。連続するドット間の空白にアクセスする無駄が省け，1回で4ドット書き込めるのは強い。8×8ドットのチップを描くときなんかはもっとも効果が出るだろう。

●特殊効果　★★★

このマシンを使うからにはやはりグラフィックの処理に関するアルゴリズムというのがもっとも興味深い。

特にスーパーファミコンで使われまくっている回転拡大縮小というのが私は好きだ。派手だが実はあまり使い道がないのがいい。いまとなっては画像処理の基本臭いが，あまりまっとうな説明を聞いた覚えがないのでここでやってしまうべきか。なんか，結局こんなことをやってしまうのは情けないが，マシン語ならでは，という処理のひとつだと思う。

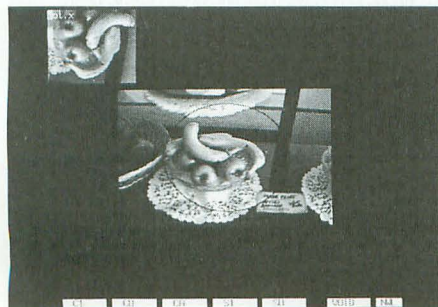
まあ，リストを見ればいろいろと頑張っている部分というものが見えるからちょうどよい材料ではないかな，と思ってもいる。ということでリストだ。まんべんなく高速化のためのコード書きが行われていると思う。もし，参考になるようならば幸いである。

具体的なことは図1に示すことにする。

こんな予定ではなかったのに

以上である。

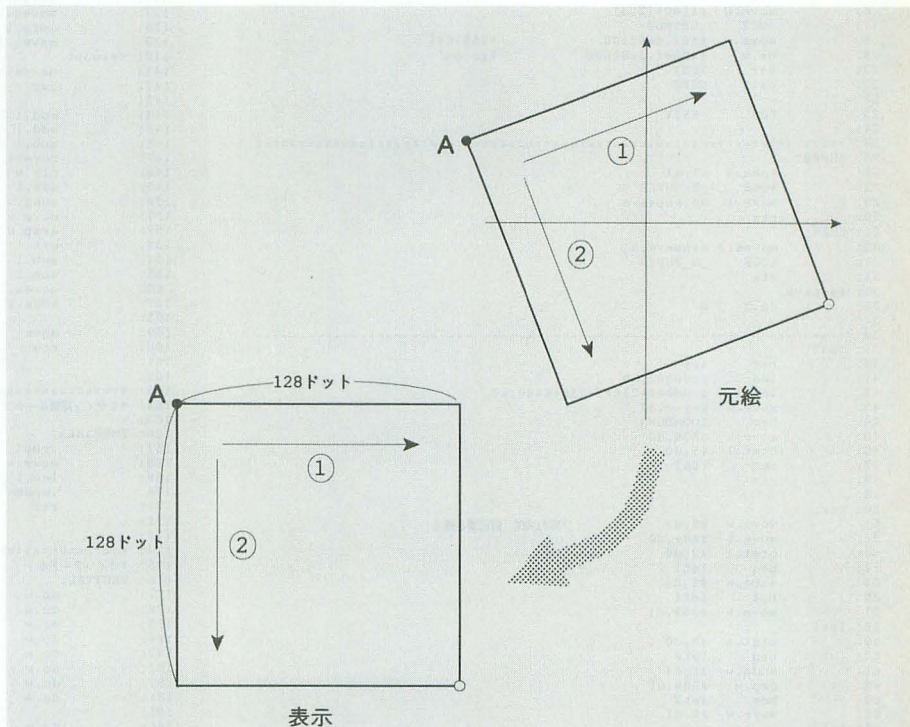
本当はもっとシブいテクニックを列挙したかったのだが，なんかこう，どの程度がテクニックかわからないレベルになってしまっているのだ。実は，テクニックとはそのままアルゴリズム構築を意味するようになっていたのかもしれない。

マシン語のコード自体での技というものは，そんなに応用のきくようなものではない。結局，マシン語でのテクニックとは，やはり最適な流れを作って無駄なストア，リストアを削り，極力レジスタだけで処理していくことではないだろうかと感じている。ほら，パチンコでもそうじゃないか，レジスタと同じように持ち玉で勝負できることがなによりも有効じゃないか（全然違うか）。

断言する。とにかくレジスタだけを使うような処理順序を作れ。細かいクロック削りはそれからでも十分。とにかくプログラムを作りまくれ。それがテクニックを磨く最高の方法だ，と思うよ。相変わらずなんかシックリこないな。

＊　＊　＊

アキラも面白いけどリオンもよい。よろめかせて穿弓腿，浮かせて後蹴腿なんかが決め技。前にかわして投げなんてみっともないからやんないよ。

画像の回転

図1

サンプルでは（256，256）を中心とした領域の128×128ドット分の画像を回転させて表示する。OPT.1，2で左右回転，スペースキーで終了する。

回転をθとするとA点は中心を（0，0）としたときに，

$$((-64)\cos\theta+(-64)\sin\theta,\ (-)(-64)\sin\theta+(-64)\cos\theta)$$

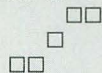
となる。一般的な座標系とはY方向の符号が逆なので注意すること。この点から①の方向に沿って画像を表示することを，縦（②の方向）の分だけ繰り返す。

まず，①の方向に沿った描画ルーチンを展開するがa0を描画先，a1を元絵のアドレスとすると，

```
move.w     (a1), (a0)+
lea        $xxx(a1), a1
```

を繰り返すことで展開する。角度によって変化するのは'$xxx'の部分なので，これを計算することによって書き換える。

たとえば，①のラインが，

```
        □□
      □
  □□
```

のようなとき，$xxxにあたる数字は順に$2，$3FE，$3FE，$2である。

これを②に沿って呼び出すことを繰り返せば表示が完成。ちなみにサンプルではアスペクト比の補正は行っていない。

注目点は，小数点の扱い方。ドットを小数で表してから計算していることである。あとは展開の方法。基本といえば基本である。そういえば自己書き換えなんかもやっている（50行）。

本当ならばもっと気違いじみたプログラムを用意すべきだろうが，視認性は最悪，説明のしようがないことが多かったのでやめた。どのへんで積極的にコスい技を使っているかというと，実はあまりしない。キーボードの読み出しは一般的な使い方ではないかもしれない。まあ，世間に出回っているプログラムよりは，まあまあの汚さで勝っているのではないだろうか。

リスト ROT.HAS

```
  1:
  2: ********************************************************************
  3: *
  4: *                 グラフィック回転デモ
  5: *
  6: ********************************************************************
  7:
  8:         .include        IOCSCALL.MAC
  9:         .include        DOSCALL.MAC
 10:
 11:         .text
 12:         .even
 13:
 14:
 15:         bsr     SUPER
 16:         move.w  #$100+12,d1
 17:         IOCS    _CRTMOD
 18:         move.w  #$03,$e82400            *65536*1
 19:         or.w    #$000f,$e82600          *gr on!
 20:         bsr     TEST
 21:         bsr     USER
 22:
 23:         DOS     _EXIT
 24:
 25: *===================================================================
 26: SUPER:
 27:         suba.l  a1,a1
 28:         IOCS    _B_SUPER
 29:         move.l  d0,sspsave
 30:         rts
 31: USER:
 32:         movea.l sspsave,a1
 33:         IOCS    _B_SUPER
 34:         rts
 35: sspsave:
 36:         dc.l    0
 37:
 38: *-------------------------------------------------------------------
 39: TEST:
 40:         bsr     key
 41:         lea.l   $c00000,a0
 42:         lea.l   $c00000+256*2+256*$400,a1
 43:         move.w  key+2,d7
 44:         bsr     ZOOMDOWN
 45:         move.b  $806,d0
 46:         btst.l  #5,d0
 47:         beq     TEST
 48:         rts
 49:
 50: key:
 51:         move.w  #0,d1                   *回転角度 自己書き換え
 52:         move.b  $80e,d0
 53:         btst.l  #2,d0
 54:         beq     let1
 55:         subq.w  #1,d1
 56:         bpl     let1
 57:         move.w  #359,d1
 58: let1:
 59:         btst.l  #3,d0
 60:         beq     let2
 61:         addq.w  #1,d1
 62:         cmp.w   #360,d1
 63:         bcs     let2
 64:         move.w  #0,d1
 65: let2:
 66:         move.w  d1,key+2
 67:         rts
 68:
 69:
 70:
 71: *==================================================================
 72: *in :a0=writeadr a1=dataadr d7=rot(0--359)
 73: *==================================================================
 74: ZOOMDOWN:
 75: *
 76:         lea.l   VECTTBL(pc),a2          *サインテーブル
 77:         lea.l   90*2(a2),a3             *コサインテーブル
 78:         add.w   d7,d7                   *テーブル1個=2バイト
 79:         move.w  (a3,d7.w),d1
 80:         move.w  0(a2,d7.w),d0           *(00----$4000)
 81:         ext.l   d0
 82:         ext.l   d1
 83:         lsl.l   #2,d0           *0~$10000
 84:         lsl.l   #2,d1           *下位ワードは小数
 85:         movea.l d0,a2           *sin
 86:         movea.l d1,a3           *cos
 87:
 88: *まず始点を計算する
 89:         move.l  d0,d2           *d2=sin
 90:         lsl.l   #6,d2           *sin
 91:         lsl.l   #6,d1           *cos 64ドット分
 92:         move.l  d1,d0
 93:         add.l   d2,d0           *x'=-(Xcos+Ysin)
 94:         neg.l   d0
 95:         sub.l   d2,d1           *y'=-(-Xsin+Ycos)
 96:         neg.l   d1
 97:
 98:         swap.w  d0              *整数部
 99:         clr.w   d1              *小数部クリア
100:         asr.l   #6,d1           *Y整数部×$400
101:         ext.l   d0              *符号の関係で、ロングワードにする
102:         add.l   d0,d1
103:         add.l   d0,d1
104:         adda.l  d1,a1           *始点アドレス
105: move.l  #0,(a1)
106: *横1ライン(128ドット)分の描画ルーチンを作る
107:
108:         lea.l   ZMPRAREA+4(pc),a4       *オフセット部をポイント
109:         move.l  #$8000,d0               *X座標(0.5)
110:         move.l  d0,d1                   *Y座標(0.5)
111:         moveq.l #0,d3                   *前回の位置との差分
112:         moveq.l #127,d2                 *ループカウンタ
113: zxloop:
114:         add.l   a3,d0           *X'=X+1・cos
115:         sub.l   a2,d1           *Y'=Y+(-1)・sin
116:
117:         move.l  d1,d4           *D5=Y
118:         clr.w   d4              *小数部クリア
119:         asr.l   #6,d4           *Y1ドット=$400
120:         swap.w  d0              *X座標整数部
121:         add.w   d0,d4
122:         add.w   d0,d4           *X1ドット=2バイト
123:         swap.w  d0
124:         move.l  d4,d5
125:         sub.l   d3,d4           *前回の位置との差分
126:         move.w  d4,(a4)
127:         move.l  d5,d3           *D6=今回の位置
128:
129:         addq.l  #6,a4           *1ドット分の命令=6バイト
130:         dbra    d2,zxloop       *128ドット分ループ
131: *
132: *描画ループ
133: *
134:         lea.l   ZMPRAREA(pc),a4 *1ライン描画ルーチン
135:         move.l  a1,d7           *元絵アドレス
136:         moveq.l #0,d0           *Y
137:         moveq.l #0,d1           *X
138:         move.l  a0,d4           *描画アドレス
139:         moveq.l #127,d5         *ループカウンタ
140: zyloop:
141:         movea.l d4,a0           *描画アドレス
142:         jsr     (a4)
143:
144:         add.l   #$400,d4        *1ライン下
145:         add.l   a3,d0           *元絵の先頭Y=+cos
146:         sub.l   a2,d1           *元絵の先頭X=+sin
147:         move.l  d0,d3
148:         clr.w   d3              *Y座標小数部クリア
149:         asr.l   #6,d3           *1ライン=$400
150:         swap.w  d1
151:         move.w  d1,d2           *X座標整数部
152:         swap.w  d1
153:         ext.l   d2              *符号の関係でロングワード
154:         sub.l   d2,d3
155:         sub.l   d2,d3
156:         movea.l d7,a1
157:         adda.l  d3,a1           *元絵の先頭計算終了
158:
159:         dbra    d5,zyloop
160:         rts
161:
162:
163: *=====================================================
164: *1ライン描画ルーチン
165:
166: ZMPRAREA:
167:         .rept   128
168:         move.w  (a1),(a0)+
169:         lea.l   $ffff(a1),a1
170:         .endm
171:         rts
172:
173:
174: *=====================================================
175: *サインテーブル
176: VECTTBL:                                        *sin 0---450(451 numbers)
177:         dc.w    $0000,$011d,$023b,$0359,$0476,$0593,$06b0,$07cc
178:         dc.w    $08e8,$0a03,$0b1d,$0c36,$0d4e,$0e65,$0f7b,$1090
179:         dc.w    $11a4,$12b6,$13c6,$14d6,$15e3,$16ef,$17f9,$1901
180:         dc.w    $1a07,$1b0c,$1c0e,$1d0e,$1e0b,$1f07,$2000,$20f6
181:         dc.w    $21ea,$22db,$23c9,$24b5,$259e,$2684,$2766,$2846
182:         dc.w    $2923,$29fc,$2ad3,$2ba5,$2c75,$2d41,$2e09,$2ece
183:         dc.w    $2f8f,$304d,$3106,$31bc,$326e,$331c,$33c6,$346c
184:         dc.w    $350e,$35ac,$3646,$36db,$376c,$37f9,$3882,$3906
185:
186:         dc.w    $3985,$3a00,$3a77,$3ae9,$3b56,$3bbf,$3c23,$3c83
187:         dc.w    $3cde,$3d34,$3d85,$3dd1,$3e19,$3e5c,$3e99,$3ed2
188:         dc.w    $3f07,$3f36,$3f60,$3f85,$3fa6,$3fc1,$3fd8,$3fe9
189:         dc.w    $3ff6,$3ffd,$4000,$3ffd,$3ff6,$3fe9,$3fd8,$3fc1
190:         dc.w    $3fa6,$3f85,$3f60,$3f36,$3f07,$3ed2,$3e99,$3e5c
191:         dc.w    $3e19,$3dd1,$3d85,$3d34,$3cde,$3c83,$3c23,$3bbf
192:         dc.w    $3b56,$3ae9,$3a77,$3a00,$3985,$3906,$3882,$37f9
193:         dc.w    $376c,$36db,$3646,$35ac,$350e,$346c,$33c6,$331c
194:
195:         dc.w    $326e,$31bc,$3106,$304d,$2f8f,$2ece,$2e09,$2d41
196:         dc.w    $2c75,$2ba5,$2ad3,$29fc,$2923,$2846,$2766,$2684
197:         dc.w    $259e,$24b5,$23c9,$22db,$21ea,$20f6,$2000,$1f07
198:         dc.w    $1e0b,$1d0e,$1c0e,$1b0c,$1a07,$1901,$17f9,$16ef
199:         dc.w    $15e3,$14d6,$13c6,$12b6,$11a4,$1090,$0f7b,$0e65
200:         dc.w    $0d4e,$0c36,$0b1d,$0a03,$08e8,$07cc,$06b0,$0593
201:         dc.w    $0476,$0359,$023b,$011d,$0000,$fee3,$fdc5,$fca7
202:         dc.w    $fb8a,$fa6d,$f950,$f834,$f718,$f5fd,$f4e3,$f3ca
203:
204:         dc.w    $f2b2,$f19b,$f085,$ef70,$ee5c,$ed4a,$ec3a,$eb2a
205:         dc.w    $ea1d,$e911,$e807,$e6ff,$e5f9,$e4f4,$e3f2,$e2f2
206:         dc.w    $e1f5,$e0f9,$e000,$df0a,$de16,$dd25,$dc37,$db4b
207:         dc.w    $da62,$d97c,$d89a,$d7ba,$d6dd,$d604,$d52d,$d45b
208:         dc.w    $d38b,$d2bf,$d1f7,$d132,$d071,$cfb3,$cefa,$ce44
209:         dc.w    $cd92,$cce4,$cc3a,$cb94,$caf2,$ca54,$c9ba,$c925
210:         dc.w    $c894,$c807,$c77e,$c6fa,$c67b,$c600,$c589,$c517
211:         dc.w    $c4aa,$c441,$c3dd,$c37d,$c322,$c2cc,$c27b,$c22f
212:
213:         dc.w    $c1e7,$c1a4,$c167,$c12e,$c0f9,$c0ca,$c0a0,$c07b
214:         dc.w    $c05a,$c03f,$c028,$c017,$c00a,$c003,$c000,$c003
215:         dc.w    $c00a,$c017,$c028,$c03f,$c05a,$c07b,$c0a0,$c0ca
216:         dc.w    $c0f9,$c12e,$c167,$c1a4,$c1e7,$c22f,$c27b,$c2cc
217:         dc.w    $c322,$c37d,$c3dd,$c441,$c4aa,$c517,$c589,$c600
218:         dc.w    $c67b,$c6fa,$c77e,$c807,$c894,$c925,$c9ba,$ca54
219:         dc.w    $caf2,$cb94,$cc3a,$cce4,$cd92,$ce44,$cefa,$cfb3
220:         dc.w    $d071,$d132,$d1f7,$d2bf,$d38b,$d45b,$d52d,$d604
221:
222:         dc.w    $d6dd,$d7ba,$d89a,$d97c,$da62,$db4b,$dc37,$dd25
223:         dc.w    $de16,$df0a,$e000,$e0f9,$e1f5,$e2f2,$e3f2,$e4f4
224:         dc.w    $e5f9,$e6ff,$e807,$e911,$ea1d,$eb2a,$ec3a,$ed4a
225:         dc.w    $ee5c,$ef70,$f085,$f19b,$f2b2,$f3ca,$f4e3,$f5fd
226:         dc.w    $f718,$f834,$f950,$fa6d,$fb8a,$fca7,$fdc5,$fee3
227:         dc.w    $0000,$011d,$023b,$0359,$0476,$0593,$06b0,$07cc
228:         dc.w    $08e8,$0a03,$0b1d,$0c36,$0d4e,$0e65,$0f7b,$1090
229:         dc.w    $11a4,$12b6,$13c6,$14d6,$15e3,$16ef,$17f9,$1901
230:
231:         dc.w    $1a07,$1b0c,$1c0e,$1d0e,$1e0b,$1f07,$2000,$20f6
232:         dc.w    $21ea,$22db,$23c9,$24b5,$259e,$2684,$2766,$2846
233:         dc.w    $2923,$29fc,$2ad3,$2ba5,$2c75,$2d41,$2e09,$2ece
234:         dc.w    $2f8f,$304d,$3106,$31bc,$326e,$331c,$33c6,$346c
235:         dc.w    $350e,$35ac,$3646,$36db,$376c,$37f9,$3882,$3906
236:         dc.w    $3985,$3a00,$3a77,$3ae9,$3b56,$3bbf,$3c23,$3c83
237:         dc.w    $3cde,$3d34,$3d85,$3dd1,$3e19,$3e5c,$3e99,$3ed2
238:         dc.w    $3f07,$3f36,$3f60,$3f85,$3fa6,$3fc1,$3fd8,$3fe9
239:
240:         dc.w    $3ff6,$3ffd,$4000
241:
242:         .end
```

浮動小数点演算プロセッサの効果
Fの哲学

Taki Yasushi 瀧　康史

**X680x0シリーズでは実数演算処理は，どのように行われているのでしょう？
コプロを利用する利点はどこにあるのでしょう？
これらを理解することであなたのパソコンが何倍も速くなるかもしれません**

通称「コプロ」といわれる，浮動小数点演算プロセッサ。98やDOS/Vシリーズのようなインテル系CPUの世界では，「NDP」といわれていますが，680x0のようなモトローラ系CPUでは「FPU」といわれています。私としては，FPUという響きのほうが好きかな。2つは機能に多少違いこそあれ，どちらも同じ目的で設計されています。68000系のFPUには68881と68882の2種類があり，後者は前者の高速版で，上位コンパチブルです。

コプロ，すなわち「コプロセッサ」というと，たいていの人は「数値演算コプロセッサ」を想像しますが，本来，コプロといわれる設計システムは，数値演算に限った話ではありません。いわゆるメインCPUの構造上，付加機能をまとめたパッケージと考えるのが正解です。

たとえば68000なら，セグメント型MMUといわれる68841というコプロセッサが，68020にはデマンドページ型MMU，68851*1が用意されています。

それでも，一般には数値演算コプロセッサのことを指すことが多いので，以下本文で，「コプロ」は，無条件に数値演算コプロセッサ68881/68882を指すことにします。

このコプロはX68030にあるソケットに差し込みます。まさに別パッケージ*2なのですが，搭載することによって，まるでメインCPUに実数演算命令が追加されたかのように働きます。必要な人だけコプロを購入すれば，実数演算がかなり高速になるわけです。

＊1　もちろん，これは専用に回路を設計しなくてはならないので，MMUがほしいからといって，いまさらX68000に68841を組み込むというのは無茶な話です。

＊2　ただし，この別パッケージによる接続だと，「信号をCPU外に出さなくてはならない」「アーキテクチャを効率よく設計できない」などという理由で(これは結果的に高速化のため)，最近ではオンチップ型であることが大半です。オンチップ型とは，メインCPUの中にその機能を最初から入れてしまうということです。

68030には68851の機能縮小したPMMUがオンチップ(68EC030には入っていない)で，68040には，さらに機能縮小したPMMUと68882を機能縮小したFPUがオンチップで入っています。

最近の例ではPlayStationのCPU，R3100(?)にはオンチップでGTEが入っているという噂も聞きます。

というわけで，速くしたいならばコプロセッサを中に入れてしまうというのが，筋ということらしいです。

なにが速くなるのか？

ところで，コプロをつけたことによる恩恵はどの程度あるのでしょう？　実数演算のベンチマークであるWHETSTONEを使用して，いくつかデータを取ってみました(表1)。Ratioの比較対象は初代X68000にfloat ver.1.00でのデータです。

X68030 Dash仕様(33MHz)でコプロがない場合，WHETSTONEが約270k[WHETS/SEC]ほど出します。コプロをつけてコンパイルしなおすと，最速で約2030k[WHETS/SEC]。だいたい，7.5倍ぐらいコプロセッサによって実数演算が高速になります。一方，整数演算が中心であるDHRYSTONEはほとんど速くならないのが事実です。

つまり，実数演算なら間違いなく高速になり，整数演算は高速化されないということになります。したがって，コプロによる高速化のキーは，どのようなアプリケーションがどのようなところで実数演算を使っているか？　ということになります。

では実際に，実数を多く使用したプログラムは，いったいどのぐらいあるのでしょう？　たとえばゲーム。スプライトを使用したアクションゲームの大半は，まず使用していないでしょう。よってコプロを接続してもまったく恩恵はありません。では，最近流行の3Dゲーム，たとえば本誌の浜崎氏が作成しているSIONシリーズなどはどうでしょう？　一見3Dのための実数演算をしているようですが，演算の大半は，固定小数点化して計算を単純にしています。

C言語で実数というと浮動小数点数のことを指します。浮動小数点は仮数部と指数部に分かれ，実際に表記するときは6.03×10^{23}などと書きます。この場合，6.03が仮数部で10^{23}が指数部です。このような浮動小数点は，どのような桁を持つ演算でも，仮数部の有効数字の分は計算値が保証されますし，指数部の限界の範囲まで値を保つことができます。その代わり，計算速度は遅くなります。そこでそれを代行する意味でコプロを接続するのです。

しかしながら，演算するデータサイズがあらかじめほぼ「予想がつき」，与える値の桁がどれもほぼ同じであるなら，小数点の位置を固定し，とある範囲内で計算を行っても誤差は大きく出ません。これらはビット数列に影響するので説明を割愛します。

ここでいいたいのはコプロを利用して浮動小数点演算するよりも，MPUが固定小数点を利用したほうが速いということです。

レイトレーシングや本誌3月号のSound Effectの特集のように，できる限り出力結果の「誤差を小さく」したい場合，浮動小数点を使わねばなりません。しかしながらリアルタイム性の高いソフトウェアの場合，出力結果の正確さよりも，速度を重要視しなくてはいけないので，このように固定小数点化するわけです。浮動小数点と固定小数点の利点と欠点を表2にまとめておきましょう。

一方，実数演算はいままでの説明どおり，精密ではありますが，有効数字である仮数部が桁あふれする計算のときに可逆にはなりません。たとえば100/33をAという浮動小数点数で確保した変数に代入し，このAを33倍したときに完全に100に戻るという保証はありません。机上の計算では答は100になりますが，コンピュータの場合，Aは割り切れない値ですから，誤差が出てしまい

ます[*3]。

したがって，アルゴリズムを設計する際，「可逆にならなくてはいけないプログラム」を必要とするときには，浮動小数点を利用しないことになります。すなわちコプロは利用しません。たとえば可逆圧縮プログラム(lhaからpic，iceなどといったもの)には利用しないことになります。

したがって，コプロを利用する必要があるプログラムは，

1) 精度が必要であるもの
2) 非可逆でもよいもの

ということになります。

コプロはよく，技術計算をする人にしか，恩恵がないといいますが，これは当たっているかもしれません。実際，アプリケーションではSX-WINDOWのEasydrawぐらいしか，高速になるものはありませんし。これとて，正確さがほとんどいらない部分は固定小数点化しているらしく(推定)，コプロを実装しても体感で1.1〜1.3倍程度しか高速化しません。浮動小数点演算と固定小数点演算の比率がものをいっているのでしょうが，ベンチマークでは7.5倍を誇った演算処理能力は，どこに消えているのでしょうか？

*3 ただし，この程度の割り切れない値は，仮数部が9.9999……になり，1.00……に桁上げされるケースが大半です。ここでの説明は便宜上のものと考えてください。

Human68kの実数処理

まず，Easydrawはなぜあまり速くならないのでしょうか？ これは，Human68kの実数演算処理の管理方法に問題があるといえます。

Human68k上のプログラムは，実数演算処理でfloat?.xを利用します。float1.xは特殊なので除外したとして，float2.x，float3.x，float4.xは利用しているハードウェア環境によって選びます。

float2.xはコプロセッサのまったくついていないシステムで利用します。どのシステムでも利用できるものなので，市販されているソフトには，たいていこれが添付されています。ver.1.*とver.2.*では速度が圧倒的に違います。

float3.xはcompactXVIまでの，CPUが68000のマシンで，数値演算プロセッサを接続したものに利用します。ただし，Xellent30(s)を導入した場合はfloat4.xが利用できます。

float4.xは，X68030のようにCPUが68030で68882が使用できるマシンで動作します。

このように3種類の環境でそれぞれ違うfloat?.xがあるわけです。いちばん速いのはfloat4.xで，次いでfloat3.x，最後にfloat2.xです。

アプリケーションで実数演算する場合，float?.xで拡張されるファンクションコール($FE??コール)を使って実数演算を行います。floatを変えるだけで，それまで使用していたプログラムが一挙に高速になるというのが「理想」だったようですが，現実はそうではありませんでした。計算時間よりも，明らかにFE〜のファンクションに分岐したりする時間が大幅に取られ，結果，最高速であるはずのfloat4.xをもってしても，float2.xよりベンチマークで2倍弱しか速くならないというオチが待っていたのです。もっとも，基準となるfloat2.xがかなりがんばっているからという話があるんですが……。

これではコプロは本来の性能を生かすことができません。そこでアプリケーション側で直接コプロセッサを呼び出す方法を使用することにします。

常駐するfloat?.xがマシンの種類によって違うように，この方法はマシンによって方法が違います。それらを以下に述べてみました。

1) CPUが68000のとき

実は，MC68000に数値演算コプロセッサはハードウェア上接続できません。したがって，I/Oバスラインを使用して，コプロセッサのプロトコル(CPUとFPUの通信信号)をエミュレーションしなくてはいけません。本来MPUが自動で処理することをソフトで行うので，かなり時間を取られます。68881/2の上では計算済みでも，MC68000からデータを取りにこないという感じで，コプロセッサの機能を完全に生かすことができません。結局，68881と68882の両方が接続できますが，ほとんど速度差は出ません。

コプロセッサプロトコルのエミュレーションを，いちから書いていくのは非常に面

表2 浮動小数点と固定小数点

	浮動小数点	固定小数点
表現できる値の範囲	約10^{-307}〜10^{+308} (64bit)	32bitにおいて，約10桁の数
有効数字	15桁	10桁
処理速度	遅い	速い

表1 whetstone benchmark 結果

ちょうどよい純正マシンなどがなかったため，あまり参考にならないかもしれないが。

machine	fpu	float	Program	Ratio	k whets/sec
040turbo35MHz(cb)		float040＋pfloat	whet040f	187.95	2564.10
040turbo35MHz(cb)		float040＋pfloat	whet040	184.63	2518.89
040turbo35MHz(wt)		float040＋pfloat	whet040f	150.82	2057.61
040turbo35MHz(wt)		float040＋pfloat	whet040	147.47	2012.07
040turbo35MHz(wt)		float040＋pfloat	whet030f	138.30	1886.79
040turbo35MHz(cb)		float040＋pfloat	whet	102.63	1400.56
040turbo35MHz(wt)		float040＋pfloat	whet030	70.21	957.85
040turbo35MHz(wt)		float040＋pfloat	whet	69.81	952.38
X68030 Dash	68882	float4.x v1.02	whet040f	148.68	2028.40
X68030 Dash	68882	float4.x v1.02	whet030f	140.96	1923.08
X68030 Dash	68882	float4.x v1.02	whet040	138.56	1890.36
X68030 Dash	68882	float4.x v1.02	whet	37.47	511.25
X68030 Dash	68882	float4.x v1.02	whet030	36.93	503.78
X68030 Dash		float2.x v2.03	whet	19.76	269.69
X68030 Dash		float2.x v2.03	whet030	19.48	265.75
Xellnt30(lh)	68882	float4.x v1.02	whet040f	117.46	1602.56
Xellnt30	68882	float4.x v1.02	whet040f	100.96	1377.41
Xellnt30	68882	float4.x v1.02	whet030f	100.68	1373.63
Xellnt30	68882	float4.x v1.02	whet040	88.00	1200.48
Xellnt30s	68882	float4.x v1.02	whet030f	72.88	994.29
Xellnt30	68882	float4.x v1.02	whet030	18.27	249.25
Xellnt30s	68882	float4.x v1.02	whet030	11.66	159.06
X68000XVI24MHz	68881	float3.x v2.00	whetf	53.50	729.88
X68000XVI	68881	float3.x v2.00	whetf	37.40	510.20
X68000XVI	68881	float3.x v2.00	whet	12.23	166.94
X68000XVI		float2.x v2.03	whet	6.43	87.77
X68000SUPER		float2.x v2.03	whet	3.86	52.68
X68000初代		float2.x v1.00	whet	1.00	13.64

Ratio = (whets/sec)/13.64;
初代X68000でfloat2.x ver1.00を利用したときの値を1とする

gcc version 1.28 Tool#2(X680x0)

倒な処理です。どのくらい面倒かは，本誌1992年9月号の私の記事を参照してください。

実際には，どの命令でもだいたい同じような処理をしますから，これらはライブラリ化するのが妥当です。libcではfppp.xというアセンブラフィルタを通すことにより，コプロセッサ命令をコンパイルすることができます。fppp.xを利用すれば，コプロとして使用できないはずのMC68000で，あたかもコプロとして接続できているかのようにプログラムを書くことができるのです。ただしソースレベルなので，デバッガで追うときには涙が出るような作業をしなくてはなりませんが……。

それでも，I/O接続された数値演算プロセッサを直接操作することにより，WHETSTONEがXVIで510k[WHETS/SEC]出すことができます。たとえ数値演算プロセッサを利用しても，float3.xを使用すると167k[WHETS/SEC]しか出ないのですから，これは大きな差といってよいでしょう。

なお，こういったI/O接続では，「数値演算コプロセッサ」という表記は正確ではないので，特に「数値演算プロセッサ」と区別して呼ぶことがあります。

2）CPUが68030のとき

X68030の場合，コプロは完全にコプロとして動作します。プログラムはxgccによって，完全にコプロ命令を利用できるようになります。

X68030 Dash仕様で，float2.x ver2.02使用時，270k[WHETS/SEC]。float4.xを通してコプロを使用すると511k[WHETS/SEC]しかいきませんが，直接コプロセッサの命令を吐き出すと，2028k[WHETS/SEC]の速度を出すことができます。これは，コプロセッサ使用前の7.5倍の速度です。

Xellent30(s)の場合も同じバイナリが使えます。

Cのソースの最適化

ここまでの話で，伝説のように互換性が高かったX680x0シリーズでも，機種ごとにオプティマイズを変えたほうがよいことがわかるでしょう。特に68030＋68882のシステムは，ほかのシステムと一線を画しています。

まずこの表1の詳しい見方をお話しします。

machineというのは，使用したコンピュータです。040turbo35MHzというのはX68030-35MHz仕様のマシンに040turboを搭載したコンピュータを指します。(cb)はコピーバックキャッシュ利用時，(wt)はライトスルーキャッシュです。当然，コプロセッサはサブセット(縮小版)がオンチップで入っているのでfpuのコラムに記述はありません。

X68030 DashはX68030-33MHz仕様です。キャッシュは命令/データともにONでベンチマークテストしています。

Xellent30はX68000XVIの無改造機に搭載，Xellent30sはX68000SUPERの無改造機に搭載しています。

programというのは，プログラムの名前です。実はこのプログラムは，すべてひとつのソースです。WHETSTONEのヘッダ部に，リスト1のようなことが記述されていて，リスト2のようなmakefile(私はあまりmakefileを書くのがうまくありません)でコンパイルしています。ただ，fppp.xを利用したwhetfはX68000＋68881/2のシステムでしかコンパイルできないようです。

それでは以下にそれぞれのプログラムの説明を書きましょう。

1）whet

もっとも基本的な方法でコンパイルしたものです。すべてのマシンで動き，互換性は確実です。

2）whet030

コプロセッサのない68030に最適化するようにコンパイルしているものです。gccのオプションに-m68020を追加します。なぜかWHETSTONEの場合，whet030のほうがwhetよりも遅いようです。本来，速くならなくてはいけないはずなのですが，なぜなのでしょうか？

3）whet030f

コプロセッサのついている68030マシンに対して最適化したものです。gccのオプションに，-m68020 -m68881を追加します。さらにソース中でmath.hをincludeする前に，

```
#define __DIRECT_FPU__
```

と宣言しているため，数学関数はコプロセッサ命令に展開されます。

4）whet040

gccのオプションに-m68040を設定しただけのものです。68040にはサブセットのコプロが入っていますが，このサブセットのコプロセッサには実数演算の四則演算程度しかないため，math.hの数学関数を展開しないほうがよいかな？　と考えたのですが，ベンチマークでは下のwhet040fのほうが速かったようです。

5）whet040f

gccのオプションに-m68040を与え，プログラム中，math.hをインクルードする前に，__DIRECT_FPU__を宣言しているものです。

040と書いてあるのに030マシンで動いて

GCCにおける数値演算指定のまとめ

GCCはデフォルトで68000用のコードを出力するようにできています。しかし，GCCにはよく用いられている最適化オプションの並び以外にも，機種別に最適化するための追加指定があります。それらを使うことで互換性は失われますが，より高速なコードを出力することができます。

CPU別の指定には以下のようなものがあります。

-m68020

直接数値演算とは関係ないが，MC68020/30用に最適化されたオブジェクトを出力する。68000マシンでは実行できなくなる。

-m68881

68881/2をコプロセッサとして接続したときのためのオブジェクトを出力する。実数演算が直接行われるようになるので格段に速度が向上する。68881/2を接続していないマシンでは実行できなくなる。

-m68040

68040用に最適化されたオブジェクトを出力する。68040用となってはいるが，68030と68882という構成のマシンでも実行できる。

また，ライブラリにlibcを使用している場合には特定のシンボルを定義することによって，演算周りの処理を特定の環境用に変えることができます。

__DIRECT_IOFPU__

X68000用数値演算プロセッサボードなど，I/O接続された68881/2を直接呼び出すためのオブジェクトを出力する。数値演算プロセッサをつけないX68000やX68030では動作しない。

__DIRECT_FPU__

X68030で68882を使用した場合のオブジェクトを出力する。X68000や68882をつけないX68030では動作しない。

このシンボルを定義するには，次の2通りの方法があります。

C言語のソース中に，

```
#define ～
```

と記述しておくか，GCCの起動時に，

```
-D__DIRECT_FPU__
```

のようなオプションを加える方法です。数値演算を多量に行うプログラムではこのような処置が非常に有効になってきます。

いるのは，040よりも030のほうが命令数が多いためです。

68030＋68882のマシンでも，68040マシンでも，この最適化方法がいちばん速かったようです。

6)　whetf

CPUが68000のマシンでしかコンパイルできません。gccのオプションに-m68881を与え，プログラム中で，

#define __DIRECT_IOFPU__

と定義しています。

これによって，I/Oバス接続されたMC68881/2を直接コントロールし，高速に演算を行います。fppp.xが必要になるようです。

以上の結果，whet030，whet030fはあまり意味がないようです。WHETSTONEだけの片寄ったベンチマークテストですが，コプロセッサを接続したマシン用にwhet040fを，I/Oバス数値演算プロセッサにwhetfを，すべてのマシンで動くようにwhetをというようなスタイルで3つのバイナリを用意すると，環境ごとに最適化できるという結果が表れました。

なお，どのようなプログラムソースも，このようにコンパイルしたら，速度アップするわけではありません。プログラム中にdouble，floatなどの演算を使っている必要があり，変数の宣言がされていること，もしくはmath.hをincludeしていることが必要になります。たいていのプログラムでは関係ないでしょう。

考察

float2.x ver.2.03を利用していたときより，

●MC68000搭載機の場合

float3.xを利用	1.90倍
fppp.xを利用	5.81倍
Xellent30＋float4.x	2.83倍
Xellent30＋fpu最適化	15.7倍

●MC68030搭載機の場合

float4.xを利用	1.90倍
fpu最適化	7.71倍
040turbo＋float40.x	約5倍
040turbo＋fpu最適化 (pfloat.xを利用)	約10倍

以上のように実数演算を高速化することができます。

今回はアセンブラを使った高速化については，あえて触れないことにしました。というのも，68040の浮動小数点演算命令群は非常に少なく，C言語の吐き出すソース以上のものを書くのは，かなり難しいと思われたからです。

EX-Systemがそれぞれの数値演算環境に対応したバイナリコードをいくつかセットして発売するようなので，68030CPUやコプロセッサを持っているユーザーは快適な環境を楽しめるかもしれません。私個人としては，Easydraw ver.2でも出てくれて，コプロセッサが対応していれば……いいんですけど。

ねぇ。

リスト1

```
PROGRAM : whet040f.x whet040.x whet030f.x whet030.x whet.x wheteml.x

whet040f.x : whet.c
        gcc -o whet040f.x whet.c -O -m68040           -D__040f__ -fomit-frame-poin
ter -fstrength-reduce -fforce-mem -fforce-addr -fcombine-regs -finline-functions
 -liocs -ldos

whet040.x : whet.c
        gcc -o whet040.x whet.c -O -m68040            -D__040__ -fomit-frame-point
er -fstrength-reduce -fforce-mem -fforce-addr -fcombine-regs -finline-functions
 -liocs -ldos

whet030f.x : whet.c
        gcc -o whet030f.x whet.c -O -m68020 -m68881 -D__030f__ -fomit-frame-poin
ter -fstrength-reduce -fforce-mem -fforce-addr -fcombine-regs -finline-functions
 -liocs -ldos

whet030.x : whet.c
        gcc -o whet030.x whet.c -O -m68020            -D__030__ -fomit-frame-poin
ter -fstrength-reduce -fforce-mem -fforce-addr -fcombine-regs -finline-functions
 -liocs -ldos

whet.x : whet.c
        gcc -o whet.x     whet.c -O                   -D__000__ -fomit-frame-poin
ter -fstrength-reduce -fforce-mem -fforce-addr -fcombine-regs -finline-functions
 -liocs -ldos
```

リスト2

```
     1: /*
     2: From hplabs!sdcrdcf!sdcsvax!dcdwest!ittatc!decvax!mcnc!rti-sel!scirtp!df
h Sun Aug 25 12:55:29 1985
     3: Relay-Version: version B 2.10.2 9/18/84; site amdahl.UUCP
     4: Posting-Version: version B 2.10.2 9/5/84; site scirtp.UUCP
     5: Path: amdahl!hplabs!sdcrdcf!sdcsvax!dcdwest!ittatc!decvax!mcnc!rti-sel!s
cirtp!dfh
     6: From: dfh@scirtp.UUCP (David F. Hinnant)
     7: Newsgroups: net.sources
     8: Subject: Whetstone Benchmark source in C - enclosed
     9: Message-ID: <353@scirtp.UUCP>
    10: Date: 25 Aug 85 19:55:29 GMT
    11: Date-Received: 27 Aug 85 08:15:18 GMT
    12: Distribution: net
    13: Organization: SCI Systems, Research Triangle Park, NC
    14: Lines: 252
    15:
    16: Enclosed below is a C translation of the famous "Whetstone Benchmark"
    17: from the original Algol version.  I have inserted printf()'s as a
    18: compiler option.  I think this translation is accurate.  The only
    19: numbers I have to compare with are from an old Ridge-32 machine, and
    20: these are from a Pascal translation (I caught one error in their
    21: translation).  If anyone has any nunbers from FORTRAN, Pascal, or Algol
    22: versions of the Whetstone, I would very much like to see them.
    23:
    24:                             David Hinnant
    25:                             SCI Systems, Inc.
    26:                             {decvax, akgua}!mcnc!rti-sel!scirtp!dfh
    27:
    28:
    29: P.s., there is a .signature file at the end of the listing. */
    30:
    31: /*
    32:  *      Whetstone benchmark in C.  This program is a translation of the
    33:  *      original Algol version in "A Synthetic Benchmark" by H.J. Curnow
    34:  *      and B.A. Wichman in Computer Journal, Vol  19 #1, February 1976.
    35:  *
    36:  *      Used to test compiler optimization and floating point performanc
e.
    37:  *
    38:  *      Compile by:                 cc -O -s -o whet whet.c
    39:  *      or:                         cc -O -DPOUT -s -o whet whet.c
    40:  *      if output is desired.
    41:  */
    42: /*
    43:         このプログラムはlibc用に修正したものです(滝)
    44: */
    45:
    46:
    47: #ifdef  __040f__
    48:  #define         __DIRECT_FPU__                    // 68030+68881/2か68040用
    49:  #define         _TRGMES_        "for X68030 + 040turbo(direct FPU)¥n"
    50: #endif
    51:
    52: #ifdef  __040__
    53:  #define         _TRGMES_        "for X68030 + 040turbo¥n"
    54: #endif
    55:
    56: #ifdef  __030f__
    57:  #define         __DIRECT_FPU__                    // 68030+68881/2か68040用
    58:  #define         _TRGMES_        "for X68030 + MC68881/2¥n"
    59: #endif
    60:
    61: #ifdef  __030__
    62:  #define         __DIRECT_FLOAT__                  // float?.xをコールする
    63:  #define         _TRGMES_        "for X68030¥n"
    64: #endif
    65:
    66: #ifdef  __000__
    67:  #define         __DIRECT_FLOAT__                  // float?.xをコールする
    68:  #define         _TRGMES_        "for X680x0 series ¥n"
    69: #endif
    70:
    71: #ifdef  __000f__
    72:  #define         __DIRECT_IOFPU__                  // I/O接続されたFPUをコールする
    73:  #define         _TRGMES_        "for X68000 + MC68881/2"
    74: #endif
    75:
    76:
    77: //#define        POUT
    78: #define ITERATIONS       100 /* 1 Million Whetstone instructions */
    79:
    80: #include <math.h>
    81: #include <stdio.h>
    82: #include <sys/iocs.h>
    83: #include <sys/dos.h>
    84: #define TimerOn          ticks0 = _iocs_ontime()
    85: #define TimerOff         ticks1 = _iocs_ontime(); ticks+=(ticks1-ticks0)
    86:
    87: double           xx1, xx2, xx3, xx4, x, y, z, t, t1, t2;
    88: double           e1[4];
    89: int                      i, j, k, l, n1, n2, n3, n4, n6, n7, n8, n9, n10,
n11;
    90:
    91: long ticks ;
    92: long ticks0,ticks1 ;
    93: double seconds;
    94:
```

▶使用前，部屋には数百枚のFDが散乱。使用後，機能的にデータが整理され，部屋に女性を連れてきても「変な人」と思われない。なんのことかって？　MOですよMO。でも，いまでは週に1枚のペースでデータが増えてる。歴史は繰り返されるのか。

角谷　光憲(21)愛知県

```
 95: main()
 96: {
 97:         int     iii;
 98:         int     loops=1;
 99:
100:
101:         printf("¥nWhetstone Benchmark   %s¥n",_TRGMES_);
102:         printf("iteration=%d¥n",ITERATIONS);
103:
104:         ticks = 0;
105:
106:     for(iii=0;iii<loops;iii++){
107:
108:         TimerOn;
109:
110:         /* initialize constants */
111:
112:         t   =   0.499975;
113:         t1  =   0.50025;
114:         t2  =   2.0;
115:
116:         /* set values of module weights */
117:
118:         n1  =   0 * ITERATIONS;
119:         n2  =  12 * ITERATIONS;
120:         n3  =  14 * ITERATIONS;
121:         n4  = 345 * ITERATIONS;
122:         n6  = 210 * ITERATIONS;
123:         n7  =  32 * ITERATIONS;
124:         n8  = 899 * ITERATIONS;
125:         n9  = 616 * ITERATIONS;
126:         n10 =   0 * ITERATIONS;
127:         n11 =  93 * ITERATIONS;
128:
129: /* MODULE 1:  simple identifiers */
130:
131:         xx1 =  1.0;
132:         xx2 = xx3 = xx4 = -1.0;
133:
134:         for(i = 1; i <= n1; i += 1) {
135:                 xx1 = ( xx1 + xx2 + xx3 - xx4 ) * t;
136:                 xx2 = ( xx1 + xx2 - xx3 - xx4 ) * t;
137:                 xx3 = ( xx1 - xx2 + xx3 + xx4 ) * t;
138:                 xx4 = (-xx1 + xx2 + xx3 + xx4 ) * t;
139:         }
140:         TimerOff;
141: #ifdef POUT
142:         Pout(n1, n1, n1, xx1, xx2, xx3, xx4);
143: #endif
144:         TimerOn;
145:
146: /* MODULE 2:  array elements */
147:
148:         e1[0] =  1.0;
149:         e1[1] = e1[2] = e1[3] = -1.0;
150:
151:         for (i = 1; i <= n2; i +=1) {
152:                 e1[0] = ( e1[0] + e1[1] + e1[2] - e1[3] ) * t;
153:                 e1[1] = ( e1[0] + e1[1] - e1[2] + e1[3] ) * t;
154:                 e1[2] = ( e1[0] - e1[1] + e1[2] + e1[3] ) * t;
155:                 e1[3] = (-e1[0] + e1[1] + e1[2] + e1[3] ) * t;
156:         }
157:         TimerOff;
158: #ifdef POUT
159:         Pout(n2, n3, n2, e1[0], e1[1], e1[2], e1[3]);
160: #endif
161:         TimerOn;
162:
163: /* MODULE 3:  array as parameter */
164:
165:         for (i = 1; i <= n3; i += 1)
166:                 pa(e1);
167:
168:         TimerOff;
169: #ifdef POUT
170:         Pout(n3, n2, n2, e1[0], e1[1], e1[2], e1[3]);
171: #endif
172:         TimerOn;
173:
174: /* MODULE 4:  conditional jumps */
175:
176:         j = 1;
177:         for (i = 1; i <= n4; i += 1) {
178:                 if (j == 1)
179:                         j = 2;
180:                 else
181:                         j = 3;
182:
183:                 if (j > 2)
184:                         j = 0;
185:                 else
186:                         j = 1;
187:
188:                 if (j < 1 )
189:                         j = 1;
190:                 else
191:                         j = 0;
192:         }
193:         TimerOff;
194: #ifdef POUT
195:         Pout(n4, j, j, xx1, xx2, xx3, xx4);
196: #endif
197:
198: /* MODULE 5:  omitted */
199:
200:         TimerOn;
201: /* MODULE 6:  integer arithmetic */
202:
203:         j = 1;
204:         k = 2;
205:         l = 3;
206:
207:         for (i = 1; i <= n6; i += 1) {
208:                 j = j * (k - j) * (l -k);
209:                 k = l * k - (l - j) * k;
210:                 l = (l - k) * (k + j);
211:
212:                 e1[l - 2] = j + k + l;          /* C arrays are zero bas
ed */
213:                 e1[k - 2] = j * k * l;
214:         }
215:         TimerOff;
216:
217: #ifdef POUT
218:         Pout(n6, j, k, e1[0], e1[1], e1[2], e1[3]);
219: #endif
220:
221:         TimerOn;
222: /* MODULE 7:  trig. functions */
223:
224:         x = y = 0.5;
225:
226:         for(i = 1; i <= n7; i +=1) {
227:                 x = t * atan(t2*sin(x)*cos(x)/(cos(x+y)+cos(x-y)-1.0));
228:                 y = t * atan(t2*sin(y)*cos(y)/(cos(x+y)+cos(x-y)-1.0));
229:         }
230:         TimerOff;
231: #ifdef POUT
232:         Pout(n7, j, k, x, x, y, y);
233: #endif
234:
235:         TimerOn;
236: /* MODULE 8:  procedure calls */
237:
238:         x = y = z = 1.0;
239:
240:         for (i = 1; i <= n8; i +=1)
241:                 p3(x, y, &z);
242:         TimerOff;
243: #ifdef POUT
244:         Pout(n8, j, k, x, y, z, z);
245: #endif
246:
247:         TimerOn;
248: /* MODULE9:  array references */
249:
250:         j = 1;
251:         k = 2;
252:         l = 3;
253:
254:         e1[0] = 1.0;
255:         e1[1] = 2.0;
256:         e1[2] = 3.0;
257:
258:         for(i = 1; i <= n9; i += 1)
259:                 p0();
260:
261:         TimerOff;
262: #ifdef POUT
263:         Pout(n9, j, k, e1[0], e1[1], e1[2], e1[3]);
264: #endif
265:
266:         TimerOn;
267: /* MODULE10:  integer arithmetic */
268:
269:         j = 2;
270:         k = 3;
271:
272:         for(i = 1; i <= n10; i +=1) {
273:                 j = j + k;
274:                 k = j + k;
275:                 j = k - j;
276:                 k = k - j - j;
277:         }
278:         TimerOff;
279: #ifdef POUT
280:         Pout(n10, j, k, xx1, xx2, xx3, xx4);
281: #endif
282:
283:         TimerOn;
284: /* MODULE11:  standard functions */
285:
286:         x = 0.75;
287:         for(i = 1; i <= n11; i +=1)
288:                 x = sqrt( exp( log(x) / t1));
289:
290:         TimerOff;
291: #ifdef POUT
292:         Pout(n11, j, k, x, x, x, x);
293: #endif
294:
295:     }
296:
297:   seconds=(double)ticks/(double)loops/(10*ITERATIONS);
298:
299:   printf("¥nWhetstone runs in %0.2f seconds. %0.2f K whets/second¥n", se
conds, 1000.0/seconds);
300:   printf("Ratio to the first X68000 and the first float2.x : %5.3f¥n",(f
loat)73.30/seconds);
301:   exit(0);
302:
303: }
304:
305: pa(e)
306: double e[4];
307: {
308:         register int j;
309:
310:         j = 0;
311:     lab:
312:         e[0] = (  e[0] + e[1] + e[2] - e[3] ) * t;
313:         e[1] = (  e[0] + e[1] - e[2] + e[3] ) * t;
314:         e[2] = (  e[0] - e[1] + e[2] + e[3] ) * t;
315:         e[3] = ( -e[0] + e[1] + e[2] + e[3] ) / t2;
316:         j += 1;
317:         if (j < 6)
318:                 goto lab;
319: }
320:
321:
322: p3(x, y, z)
323: double x, y, *z;
324: {
325:         x  = t * (x + y);
326:         y  = t * (x + y);
327:         *z = (x + y) /t2;
328: }
329:
330:
331: p0()
332: {
333:         e1[j] = e1[k];
334:         e1[k] = e1[l];
335:         e1[l] = e1[j];
336: }
337:
338: #ifdef POUT
339: Pout(n, j, k, x1, x2, x3, x4)
340: int n, j, k;
341: double x1, x2, x3, x4;
342: {
343:         printf("%5d %5d %5d   %11.3e %11.3e %11.3e %11.3e¥n",n, j, k, x1
, x2, x3, x4);
344: }
345: #endif
346:
347: /*
348: ====================================================================
349: --
350:                                 David Hinnant
351:                                 SCI Systems, Inc.
352:                                 {decvax, akgua}!mcnc!rti-sel!scirtp!dfh
353:
354: */
```

▶Xellent30の取り付けついでに，本体をバラして掃除してやりました。意外にホコリがたまっていますね。あとは5，6カ所細工して組み立て。うーん快適。

渡辺　久孝(28)大阪府

コンパイラの挙動を知る

GCCにおける最適化

Nakamori Akira 中森 章

最適化されるのはなにもアセンブラだけではない
C言語では自動的に最適化され，そしてGCCは非常に優秀なコンパイラだ
ここでは用意されているものを有効に使うことを考えてみよう

はじめに

現在，X680x0の開発で使用されているCコンパイラはGCC (GNU CC) がほとんどです。シャープからも純正のCコンパイラであるXCが発売されていますが，XCは出力するコードの質があまりよくなく最適化もほとんど行わないので，現在ではそのライブラリ以外は見捨てられた格好になっています。

今回はX680x0の実質上の標準になっているGCCでどのようなコードの最適化が行われているか紹介しましょう。すでにGCCの最適化については何度か解説したことがありますが，XGCCの最適化は知らないでいるとかなり損をしてしまうこともあります。特に最適化オプションをつけないで使っている人にはぜひとも，また，普段は「おまじない」としてオプションを設定している人もその内容の確認の意味で読んでみてください。

最適化の実際

GCCの特長は出力するコードの最適化がよく行われているということです。コンパイラの最適化の理論は昔からエイホとウルマンなどの教科書でいくつも紹介されています。しかし，数年前まで，その昔ながらの理論ですら実際に適用しているCコンパイラというのはほとんどありませんでした（少なくとも適用しているとは思えなかった)。現在のパソコンやワークステーションの世界ではCコンパイラの性能競争が激しく，ひととおりの最適化は行われるようになっているようですが，GCCは昔から意欲的に最適化を行ってきており，この点ほかのCコンパイラに一歩先んじていました。

ここではGCCが実際に行っている最適化を紹介するわけですが，使用したCコンパイラはソフトバンク刊の「X680x0 Develop. & libc II」に収録されているバージョンです。

基本的に環境変数のGCC_OPTIONにはなにも設定せずに，最適化は，

```
gcc -S -O hoge.c
```

などとコマンドラインのみからのオプションスイッチで指定しています。

定数の畳み込み

これはコンパイル時に計算可能な式をあらかじめ計算してしまう最適化です。たとえば，

```
x = 2*3;
```

という式がある場合，実行時に常に2*3を計算してxに代入するのは時間の無駄です。それよりも式を計算して，

```
x = 6;
```

としておけばxには直接6を代入するだけでよくなります。

このように直接定数式をプログラム中に記述することはまれかもしれません。しかし，プログラムではプリプロセッサの#defineで定義された定数値と別の数値との演算は非常によく現れるのでこの最適化は重要になってきます。たとえば，

```
#define HOGE 100
```

という宣言があるとして，プログラム中に

```
x=HOGE+1;
```

とか，

```
x=30*HOGE;
```

文があるとします。このような文はプリプロセッサを通過してコンパイラに渡るときには，

```
x=100+1;
x=30*100;
```

と展開されているので定数の畳み込みの意味が大いに生きてきます。これはもっとも基本的な最適化ですからやってないコンパイラはまずないでしょう。

また，直接に式で表されていなくても，あらかじめ値を計算することで実行速度を上げられる場合もあります。たとえば，aというint型配列の，

```
a[i+1]
```

という要素を参照する場合，要素のアドレスは，

aのアドレス＋(i+1)＊4

によって計算されます。このとき

(aのアドレス＋4)＋i＊4

と変形して(aのアドレス＋4)をあらかじめ計算しておけばiに1を加えてから4倍するという操作が単純にiを4倍する操作に置き換わります。これはa[i+1]がループ内に現れるとき特に有効です。リスト1に定数の畳み込みをともなうプログラムとコンパイル結果を示します。

定数の伝播

これは，定数が代入されている変数の参照を，その定数の参照に置き換える処理です。

たとえば，

```
x=1;
y=x+2;
```

は，

```
x=1;
y=3;
```

とすることができます。リスト2に定数の伝播に関するプログラムとコンパイル結果を示します。ところで，GCCでは大域変数に対しては，通常は，定数の伝播を行わないようです。

しかし，GCCには大域変数をレジスタに割り付ける，

```
-fforce-mem
```

という（最適化の）オプションスイッチがあります。これをつけてコンパイルすると大域変数にも定数の伝播が行われるようです。

単純代入の削除

　a＝b；
のような演算のない代入を単純代入といいます。変数aとbは同じ値を持つため，aの参照をbの参照に置き換えてしまうことで，
　a＝b；
という代入文を削除できる場合があります。たとえば，
　a＝b；
　x＝a＋c；
　y＝a−c；
は，
　x＝b＋c；
　y＝b−c；
と置き換えることができます。通常，このような単純代入文を記述することはありませんが，さまざまな最適化の過程で生じることが多いそうです。したがって，単純代入文の削除は最適化においては必須な技術になっています。リスト3に単純代入文を削除するプログラムの例とコンパイル結果を示します。

無駄コードの削除

よく考えずにプログラムを書くと無駄なコードが発生してしまうことがあります。たとえば，
・return文よりも後の未到達コード
・宣言しただけで使用しない変数
・条件が常に偽のif文
・実行されないループ
などです。

これらのコードは最終的な実行結果に無関係ですから，それらに対するコードを生成することは実行速度の面からもプログラムサイズの面からも不利になります。このような無駄コードは削除されなくてはなりません。わざわざ無駄なコードを書く人はいないと思いますが，コンパイラのコード生成の仕方によっては，たまたま無駄なコードが生まれる場合もあります。

また使用しない変数を宣言することは結構あるのではないでしょうか。リスト4に無駄なコードを含むプログラムとそのコンパイル結果を示します。GCCではすべての無駄コードが削除されて非常にスッキリしたコードを生成しています。

共通部分式の削除

プログラム内では意図的にあるいは無意識に同じ式を何度か繰り返して書く場合があります。同じ値を持つ式が2カ所以上で現れる（これを共通部分式という）ならば，それらを一度に計算したほうが演算回数を減らすことができます。これが共通部分式の削除と呼ばれる最適化です。これはいわゆる最適化コンパイラならば必ず行っているもっとも代表的な最適化です。たとえば，
　x＝a＊b＋c；
　y＝a＊b−c；
という式があれば，a＊bという共通部分式を認識して，
　t＝a＊b；
　x＝t＋c；
　y＝t−c；
という式に変換するのが共通部分式の削除

リスト1

```
/*
定数の畳み込み
*/
int a[10];
int x,y,z;
foo()
{
     int i;
     x=2*3;
     y=1+2+3+4;
     for(i=0;i<9;i++)
          a[i+1]=x;
}
-----コンパイル結果----------
gcc -S -O でコンパイル
------------------------------
_foo:
          link a6,#0
          moveq.l #6,d2    ; 2*3
          move.l d2,_x
          moveq.l #10,d2   ; 1+2+3+4
          move.l d2,_y
          moveq.l #0,d1    ; i=0
          lea _a+4,a0      ; aのｱﾄﾞﾚｽ+4
?5:
          move.l d1,d0
          asl.l #2,d0      ; i=4*i
          move.l _x,(a0,d0.l) ; a[i+1]=x
          addq.l #1,d1
          moveq.l #8,d2
          cmp.l d1,d2
          jbge ?5
          unlk a6
          rts
```

リスト2

```
/*
定数の伝播（その1）
*/
int a,b,c;
foo()
{
     int x,y,z;
     x = 1;
     y = x+2;
     z = x+y;
     a = x;
     b = y;
     c = z;
}
-----コンパイル結果---------
gcc -S -O でコンパイル
-----------------------------
_foo:
          link a6,#0
          moveq.l #1,d0    ; x=1
          move.l d0,_a
          moveq.l #3,d0    ; y=x+2(=3)
          move.l d0,_b
          moveq.l #4,d0    ; z=x+y(=4)
          move.l d0,_c
          unlk a6
          rts
-----------------------------


/*
定数の伝播（その2）
*/
int x,y,z;
foo()
{
     x = 1;
     y = x+2;
     z = x+y;
}
-----コンパイル結果----------
gcc -S -O でコンパイル
------------------------------
_foo:
          link a6,#0
          moveq.l #1,d0
          move.l d0,_x
          move.l _x,d0
          addq.l #2,d0     ; x+2を計算
          move.l d0,_y
          move.l _x,d0
          add.l _y,d0      ; x+yを計算
          move.l d0,_z
?1:
          unlk a6
          rts
------------------------------
gcc -S -O -fforce-mem でコンパイル
------------------------------
_foo:
          link a6,#0
          moveq.l #1,d0
          move.l d0,_x
          moveq.l #3,d0    ; 1+2
          move.l d0,_y
          moveq.l #4,d0    ; 1+(1+2)
          move.l d0,_z
          unlk a6
          rts
```

リスト3

```
/*
単純代入の削除
*/
int x,y;
foo()
{
     int a,b,c;
     a = b;
     x = a+c;
     y = a-c;
     /* 以後 a は参照しない */
}
-----コンパイル結果-----------
gcc -S -O でコンパイル
--------------------------------
_foo:
          link a6,#0
          move.l d1,d2     ; b
          add.l d0,d2      ; b+c
          move.l d2,_x
          move.l d1,d2     ; b
          sub.l d0,d2      ; b-c
          move.l d2,_y
          unlk a6
          rts
```

リスト4

```
/*
無駄コード削除
*/
int      x;
foo()
{
     int i,j;
     j=0;                  /* 未使用変数 */
     i=0;
     if(0)    f(i);        /* 未到達コード */
     x=i;
     return;
     x+=10;                /* 未到達コード */
}
-----コンパイル結果---------
gcc -S -O でコンパイル
-----------------------------
_foo:
          link a6,#0
          moveq.l #0,d0    ; i=0
          move.l d0,_x     ; x=i
          unlk a6
          rts
```

です。この変換で代入の回数は増えてしまいましたが演算回数（この場合は実行の遅い乗算の回数）を減らすことができました。リスト5に共通部分式の現れるプログラムとコンパイル結果を示します。共通部分式がうまく削除されているのがわかると思います。ただ，このプログラムにはトリックがあります。a+b（加算）という式をa*b（乗算）に置き換えるとGCCでも（なにもオプションスイッチをつけない限り）共通部分式の削除ができなくなってしまいます。これは乗算が関数によって実行される（MC68000に32ビット乗算命令はない）ために，一般的な最適化のアルゴリズムが使えなくなるためと考えられます（関数はいろいろな副作用をともなうのでそれを含む式の最適化は難しい）。乗算がある場合は，

-m68020

というオプションをつけてコンパイルしてみましょう。これはX68030（32ビット乗算命令があるMC68EC030を使用している）用のコードを出力するためのオプションです。この場合は乗算を含む共通部分式もちゃんと削除されるようです。

リスト5

```
/*
共通部分式の削除
*/
int     x,y,z;
int     a,b,c;
foo()
{
    x=a+b-c;
    y=a+b+c;
    z=a+b;
}
-----コンパイル結果---------
gcc -S -O でコンパイル
----------------------------
_foo:
        link a6,#0
        move.l _a,d0
        add.l _b,d0         ; a+b
        move.l d0,d1        ; (a+b)
        sub.l _c,d1         ; (a+b)-c
        move.l d1,_x
        move.l d0,d1        ; (a+b)
        add.l _c,d1         ; (a+b)+c
        move.l d1,_y
        move.l d0,_z        ; (a+b)
        unlk a6
        rts
----------------------------------------


/*
共通部分式の削除（その2）
*/
int     x,y,z;
int     a,b,c;
foo()
{
        x=a*b-c;
        y=a*b+c;
        z=a*b;
}
-----コンパイル結果---------
gcc -S -O でコンパイル
----------------------------
_foo:
        link a6,#0
        move.l a3,-(sp)
        lea ___mulsi3,a3
        move.l _b,-(sp)
        move.l _a,-(sp)
        jbsr (a3)           ; a*b
        addq.w #8,sp
        sub.l _c,d0         ; (a*b)-c
        move.l d0,_x
        move.l _b,-(sp)
        move.l _a,-(sp)
        jbsr (a3)           ; a*b 再計算
        addq.w #8,sp
        add.l _c,d0         ; (a*b)+c
        move.l d0,_y
        move.l _b,-(sp)
        move.l _a,-(sp)
        jbsr (a3)           ;  a*b 再計算
        move.l d0,_z        ; (a*b)
        move.l -4(a6),a3
        unlk a6
        rts
-----------------------------
gcc -S -O -m68020 でコンパイル
-----------------------------
_foo:
        link a6,#0
        move.l _a,d0
        muls.l _b,d0        ; a*b
        move.l d0,d1        ; (a*b)
        sub.l _c,d1         ; (a*b)-c
        move.l d1,_x
        move.l d0,d1        ; (a*b)
        add.l _c,d1         ; (a*b)+c
        move.l d1,_y
        move.l d0,_z        ; (a*b)
        unlk a6
        rts
```

ループ内不変式の移動

ループ内にあって，値がループの繰り返しによっても変化しない式をループ内不変式といいます。このような式をループの繰り返しごとに計算し直すのは時間の無駄です。そこでループ内不変式はループに入る前に計算してしまうという考え方があります。これがループ内不変式の移動と呼ばれる最適化です。たとえば，

```
for(i=0;i<10;i++){
    x=x+a;
    a=b-c+d;
}
```

というループがあるとき，(b−c+d)の値はループ内で不変です。このような場合は，

リスト6

```
/*
ループ内不変式の移動
*/
int     x;
int     a,b,c,d;
foo()
{
    int i;
    a=1;
    for(i=0;i<10;i++){
        x=x+a;
        a=b-c+d;        /* 不変式 */
    }
}
-----コンパイル結果----------
gcc -S -O でコンパイル
------------------------------
_foo:
        link a6,#0
        moveq.l #1,d2
        move.l d2,_a        ; a=1
        moveq.l #0,d1       ; i=0
        move.l _b,d0
        sub.l _c,d0
        add.l _d,d0         ; b-c+d
?5:
        move.l _a,d2
        add.l d2,_x         ; x=x+a
        move.l d0,_a        ; a=(b+c-d)
        addq.l #1,d1
        moveq.l #9,d2
        cmp.l d1,d2
        jbge ?5
        unlk a6
        rts
```

```
t=b-c+d;
for(i=0;i<10;i++){
    x=x+a;
    a=t;
}
```

としてコンパイルすればループ内での計算回数を減らすことができます。リスト6にループ内不変式を持つプログラムとそのコンパイル結果を示します。なお，リスト1のコンパイル結果で，ループ外で(aのアドレス+4)をa0レジスタに入れていたのもこのループ内不変式の移動です。

自動変数のレジスタ割り付け

C言語では使用頻度の高い変数を高速に参照するために（できるだけ）レジスタに割り付けるためのregister宣言がありました。しかし，そのようなことは人間がいちいち指示しなくてもコンパイラが自主的に行うのが正しいあり方です。

GCCでは自動変数（局所変数）をできるだけレジスタに割り付けるようなコード生成を行います。そのため，自動変数を使用する式を高速に計算することができます。静的変数（大域変数）は通常はレジスタに割り付けられませんがコンパイル時のオプションスイッチ（-fforce-mem）でレジスタに割り付けることも可能です。

一般にRISC用のCコンパイラが非常に性能がいいのは（ひととおりの最適化のほかに）大域変数のレジスタ割り付けがよく行われているからです。GCCでは大域変数のレジスタ割り付けはほんのオマケといった程度でしかなく，中途半端にレジスタ割り付けが行われるためか性能が低下する場合もあります。リスト7にレジスタ割り付けを行う例を示します。

連続関数呼び出し時の引数ポップ

通常のCコンパイラで，

f(a,b);

という関数呼び出しをコンパイルすると次のようなコードが生成されます。

```
push b      ;引数をスタックに積む
push a      ;引数をスタックに積む
jbsr f      ;関数を呼ぶ
add  #8,sp  ;スタックを補正する
```

つまり，引数のプッシュによってずらしたスタックポインタを関数から戻ってきた後で補正します。ところで，もし関数の呼び出しが，

f(a,b);

```
	g(x,y);
```
のように連続する場合は，

```
	push b      ;引数をスタックに積む
	push a      ;引数をスタックに積む
	jbsr f      ;関数を呼ぶ
	add #8,sp   ;スタックを補正する
	push y      ;引数をスタックに積む
	push x      ;引数をスタックに積む
	jbsr g      ;関数を呼ぶ
	add #8,sp   ;スタックを補正する
```
となりますが，この命令列をよく眺めると，

```
	jbsr f
```
と，

```
	push y
```
の間にある，

```
	add #8,sp
```
はあまり意味のないコードであるとわかります。このコードを省略すると関数gを呼ぶ前にスタックポインタの値が8だけずれたままになります。しかし，gから戻ってきた後に，

```
	adda.l #16,sp
```
によって一括してスタックポインタの補正をしてやればなにも不都合は起きません。GCCはこういう方法で関数呼び出し後のスタックポインタの補正を省略して実行速度を上げています。これを示すプログラムがリスト8です。

なお，リスト8のコンパイル結果では，unlk命令でスタックポインタが関数が呼び出された直後の値に復帰することを利用して，最終的なスタックポインタの補正まで省略しています(素晴らしい)。もっとも常にこんな最適化を行うと不都合が生じる場合があるかもしれないので，GCCではこのようなスタックポインタの一括しての補正を禁止するコンパイル時のオプションスイッチ(-fno-defer-pop)も用意されています。これで万全ですね。

ところで，リスト8では最初の部分で先に述べた定数の伝播が行われているのがわかると思います。

ループの最適化

ループ内で配列要素を参照するときそれがループごとに連続した領域になることがしばしばあります。たとえば，

```
	for(i=0;i<100;i++)
		a[i]=0;
```
というint型の配列を初期化するループを考えてみましょう。この場合，ループ内にあるa[i]という配列要素のアドレスは，

```
	aのアドレス+i*sizeof(int)
```
によって計算されます。つまり1回の乗算と1回の加算が必要です。しかし，ポインタ変数を用いた，

```
	p=&a[0];
	for(i=0;i<100;i++){
		*p=0;
		p++;
	}
```
というループで同じことを行う場合は配列要素のアドレスをわざわざ計算する必要はありません。p++によって次の要素のアドレスを計算してはいますが，これは加算1回で行うことができます(MC680x0のポストインクリメントアドレッシングを用いればその必要もない)。これはループ内の計算の「強さ」を減少したことになります(当然実行速度は上がる)。これは演算強度の軽減(ストレングスレデュース)と呼ばれる最適化です。GCCは通常はこのような最適化は行いませんがコンパイル時に，

```
	-fstrength-reduce
```
というオプションスイッチをつけることで演算強度の軽減を行うことができます。リスト9に演算強度の軽減を利用したループの最適化を行うプログラム例を示します。リスト9ではループの1回の繰り返しごとに配列要素を示すポインタの値が4ずつ増えていくので少し複雑なコードになっています。

また，ループは通常，

```
?5:
	:
	cmp.l d1,d2
	jbge ?5
```
というコードに展開されます。しかし，ループの内容がループの制御変数に無関係な場合は，

```
?5:
	:
	dbra d0,?5
```
というように，ループ専用命令である，

リスト7

```
/*
自動変数のレジスタ割り付け
*/
foo()
{
    int a,b,c,d,e;
    a=b;
    if(b==e){
        c=a+e;
        d=b;
    }
    else {
        c=a-e;
        d=e;
    }
    b=c+e;
    e=b+d;
    f(a+b+c+d+e);
}
-----コンパイル結果---------
gcc -S -O でコンパイル
----------------------------
_foo:
        link a6,#0
        move.l d3,-(sp)
        move.l d0,d3       ; a=b
        cmp.l d0,a1        ; b←d0,e←a1
        jbne ?2
        move.l d0,d2       ; c=b(c=a)
        add.l a1,d2        ; c=b+e(c=a+e)
        move.l d0,d1       ; d=b
        jbra ?3
?2:
        move.l d3,d2       ; c=a
        sub.l a1,d2        ; c=a-e
        move.l a1,d1       ; d=e
?3:
        move.l d2,d0       ; b=c
        add.l a1,d0        ; b=c+e
        move.l d0,a1       ; e=b
        add.l d1,a1        ; e=b+d
        move.l d3,a0       ; +a
        add.l d0,a0        ; +b
        add.l d2,a0        ; +c
        add.l d1,a0        ; +d
        pea (a1,a0.l)      ; +e
        jbsr _f
        move.l -4(a6),d3
        unlk a6
        rts
-----------------------------------


/*
大域変数のレジスタ割り付け
*/
int x,y,z;
foo()
{
    x = y+1;
    z = x+y;
}
-----コンパイル結果---------
gcc -S -O -fforce-mem でコンパイル
----------------------------
_foo:
        link a6,#0
        move.l _y,d0       ; y
        move.l d0,d1       ; tmp=y
        addq.l #1,d1       ; tmp=y+1
        move.l d1,_x       ; x=tmp(x←d1)
        add.l d0,d1        ; tmp=x+y
        move.l d1,_z       ; z=tmp
        unlk a6
        rts
```

リスト8

```
/*
連続関数呼び出し時の引数ポップ
*/
foo()
{
    int a,b,c;
    a=1;
    b=a+1;
    c=b+1;
    f(a,b);
    g(b,c);
    h(a,b,c);
}
-----コンパイル結果---------
gcc -S -O でコンパイル
----------------------------
_foo:
        link a6,#0
        movem.l d3/d4/d5,-(sp)
        moveq.l #1,d3      ; a=1
        moveq.l #2,d4      ; b=a+1
        moveq.l #3,d5      ; c=b+1
        move.l d4,-(sp)    ; push b
        move.l d3,-(sp)    ; push a
        jbsr _f            ; f(a,b)
        move.l d5,-(sp)    ; push c
        move.l d4,-(sp)    ; push b
        jbsr _g            ; g(b,c)
        move.l d5,-(sp)    ; push c
        move.l d4,-(sp)    ; push b
        move.l d3,-(sp)    ; push a
        jbsr _h            ; h(a,b,c)
        movem.l -12(a6),d3/d4/d5
        unlk a6
        rts
```

dbra

を使用して速度を稼ぐことがあります。この最適化も，

-fstrength-reduce

によって行われます。

フレームポインタの削除

関数の入り口では通常のCコンパイラではスタックフレームを生成し（MC680x0ではlink命令を使う），フレームポインタからの相対位置を指定して引数や局所変数を参照します。たとえば，

```
f(x,y)
int x,y;
{
    return(x+y);
}
```

という関数はMC680x0用のCコンパイラでは，

```
f:
    link a6,#0
                ;フレームポインタをa6に
    move.l 8(a6),d0 ;xをd0に
    add.l 12(a6),d0 ;yをd0に加算
    unlk a6    ;スタックフレーム削除
    rts
```

とコンパイルされるでしょう。しかし，関数に局所変数がまったくない場合，link命令でフレームポインタを作らなくてもスタックポインタからの相対位置で引数を簡単に参照することができます。たとえば，上の場合，

```
f:
    move.l 4(sp),d0 ;xをd0に
    add.l  8(sp),d0 ;yをd0に加算
    rts
```

で十分用が足りてしまいます。このときlinkとunlkの2命令が節約できたことになります（時間にしてMC68000で約66クロック減少）。こんなうまい話を性能重視のGCCが見逃すはずがありません。GCCではコンパイル時に，

-fomit-frame-pointer

というオプションスイッチを指定することでスタックフレームの作成（linkとunlkの実行）をやめることができます。これを示すプログラム例がリスト10です。リスト10を見てわかるように，関数内で局所変数を必要とする場合は必ずスタックフレームが作られるようです。

関数のインライン展開

Cコンパイラの最適化の中で究極の最適化のひとつは関数のインライン展開です。これは関数の本体をそれを呼び出す位置に埋め込んでしまい，関数呼び出しの時間を節約する最適化です。確かに実行速度は向上する（MC68000で約40クロック節約）のですが，コードサイズは明らかに増加しますし，あえて関数として宣言されたものをコード内に埋め込んでもいいものかという疑問も残ります。また，分割コンパイルの前には無力になってしまいますし，関数のインライン展開はベンチマーク以外にどの程度有用であるかどうかははっきりしません。

しかし，関数のインライン展開はコンパイラ技術としては興味深いものがあります。GCCでも簡単な関数については，

-finline-functions

というオプションをつけてコンパイルすることでインライン展開ができるようになっています（簡単な関数という定義はよくわかりませんが）。

リスト11に関数のインライン展開を行うプログラム例を示します。本当に関数が見事に埋め込まれていますね。リスト11ではインライン展開したあとには定数の畳み込みの最適化がさらに行われています（引数が定数だと関数の戻り値が計算できてしまう）。

定数代入の最適化

X680x0用のGCCでは環境変数，

GCC_OPTION

に“O”が設定してあると，X680x0版で固有に拡張された最適化処理が行われるようになります。

この最適化は，これまで紹介した最適化のほとんどがC言語のソースプログラムのレベルに関係するものであるのに対して，アセンブラのソースレベルで行われるものです。

リスト9

```
/*
ループの最適化（その1）
*/
int     x;
int     a[10];
foo()
{
    int i;
    for(i=0;i<10;i++)
        a[i]=4*i-x;
}
-----コンパイル結果----------
gcc -S -O でコンパイル
-------------------------------
_foo:
        link a6,#0
        moveq.l #0,d1   ; i=0
        lea _a,a0
?5:
        move.l d1,d0
        asl.l #2,d0         ; 4*i
        move.l d0,d2
        sub.l _x,d2         ; 4*i-x
        move.l d2,(a0,d0.l) ; (a+4*i)=4*i-x
        addq.l #1,d1        ; i++
        moveq.l #9,d2
        cmp.l d1,d2
        jbge ?5
        unlk a6
        rts
-------------------------------
gcc -S -O -fstrength-reduce でコンパイル
-------------------------------
_foo:
        link a6,#0
        lea _a,a0
        moveq.l #0,d0
?5:
        move.l d0,d1    ; 4*i
        sub.l _x,d1     ; 4*i-x
        move.l d1,(a0)+
        addq.l #4,d0    ; 4*i+4
        moveq.l #36,d1
        cmp.l d0,d1
        jbge ?5
        unlk a6
        rts
-------------------------------------------


/*
ループの最適化（その2）
*/
int     x;
foo()
{
    int i;
    for(i=0;i<10;i++)
        x = 0;
}
-----コンパイル結果----------
gcc -S -O -fstrength-reduce でコンパイル
-------------------------------
_foo:
        link a6,#0
        moveq.l #9,d0   ; ﾙｰﾌﾟ回数-1
?5:
        moveq.l #0,d1
        move.l d1,_x    ; x=0
        dbra d0,?5
        ext.l d0
        unlk a6
        rts
```

定数代入の最適化はそれにあたり，いくつかの変数を同じ値で初期化するとき，サイズの大きい変数から順に初期化するようにすれば，レジスタの値を積極的に再利用する最適化が行われます。

通常，short型やchar型の定数値の代入にはイミディエートアドレシングが使用されますので，その代わりにレジスタを使用するこの最適化は，MC68000ではそれぞれの定数の初期化につき4クロックずつの時間短縮となります。リスト12にその例を示します。

おわりに

X680x0のGCCでは，これまで見てきたようにさまざまな最適化が試みられています。

「Cコンパイラを使用するよりもアセンブラを使用したほうが効率よく高速なコーディングができる」ということはよくいわれることです。しかし，GCCに限れば初心者が下手にアセンブラを使うよりもよいコードを生成することがしばしばです。プログラム開発にはアセンブラしか使用しないという人もこの機会にGCCを試してみてはどうでしょうか。

また，いかにGCCといえども完璧に最適化されたコードを出力するわけではありませんので，GCCが生成するコードをアセンブリ言語レベルで手で最適化すればもっとすぐれたコードにすることができるのではないでしょうか。大まかな部分はC言語で記述し，要となる部分だけアセンブラで念入りに最適化したコードを使用するようにするのがよいでしょう。あとは各自で研究してみてください。

●参考文献

X68k programing Series(#1)
X680x0 Develop，ソフトバンク

リスト12

```
/*
X680x0固有の最適化
*/
int a;
short b;
char c;
foo()
{
        a = 100;
        b = 100;
        c = 100;
}
------コンパイル結果----------
gcc -S -O でコンパイル
------------------------------
_foo:
            link a6,#0
            moveq.l #100,d0
            move.l d0,_a
            move.w #100,_b   ; イミディエートを使用
            move.b #100,_c   ; イミディエートを使用
            unlk a6
            rts
------------------------------
set GCC_OPTION=O
gcc -S -O でコンパイル
------------------------------
_foo:
            link a6,#0
            moveq.l #100,d0
            move.l d0,_a
            move.w d0,_b      ; レジスタを使用
            move.b d0,_c      ; レジスタを使用
            unlk a6
            rts
```

リスト10

```
/*
フレームポインタの削除
*/
f(x,y)
int x,y;
{
        return(x+y+10);
}

g(x)
int x;
{
        int a[3];
        a[1]=f(1,2);
        return(a[1]+10);
}

int         x,y;
main()
{
        x=f(100,200);
        y=g(x);
}
-----コンパイル結果----------
gcc -S -O -fomit-frame-pointer でコンパイル
--------------------------------
            .even
            .globl _f
_f:
            move.l 4(sp),d0  ; x
            add.l 8(sp),d0   ; +y
            moveq.l #10,d1
            add.l d1,d0
            rts
            .even
            .globl _g
_g:
            link a6,#-12       ; 局所変数があ
            pea 2.w            ; る場合はスタック
            pea 1.w            ; フレームを作る
            jbsr _f
            move.l d0,-8(a6)
            moveq.l #10,d0
            add.l -8(a6),d0
            unlk a6
            rts
            .even
            .xref __main
            .xdef _main
_main:
            pea 200.w
            pea 100.w
            jbsr _f
            move.l d0,_x
            move.l d0,-(sp)
            jbsr _g
            move.l d0,_y
            lea 12(sp),sp
            rts
```

リスト11

```
/*
関数のインライン展開
*/
f(x,y)
int x,y;
{
        return(x+y+10);
}

g(x)
int x;
{
        int a[3];
        a[1]=f(1,2);
        return(a[1]+x);
}

int         x,y;
main()
{
        x=f(100,200);
        y=g(x);
}
------コンパイル結果----------
gcc -S -O -finline-functions でコンパイル
--------------------------------
            .even
            .globl _f
_f:
            link a6,#0
            move.l 8(a6),d0
            add.l 12(a6),d0
            moveq.l #10,d1
            add.l d1,d0
            unlk a6
            rts
            .even
            .globl _g
_g:
            link a6,#-12
            moveq.l #13,d1      ; f(1,2)=13
            move.l d1,-8(a6)
            move.l 8(a6),d0     ; x
            add.l d1,d0         ; f(1,2)+x
            unlk a6
            rts
            .even
            .xref __main
            .xdef _main
_main:
            link a6,#-12
            move.l #310,d0      ; f(100,200)=100+200+10
            move.l d0,_x
            moveq.l #13,d1      ; f(1,2)=13
            move.l d1,-8(a6)
            add.l d1,d0         ; f(1,2)+x
            move.l d0,_y
            unlk a6
            rts
```

ローカルRAMの使い方

Xellent30を活用する

Kikuchi Isao 菊地 功

従来機種ユーザーには福音となったXellent30
ただし，そのまま使ったのではなかなか性能を発揮できない
搭載されたSRAMを生かして使いこなすことを考えよう

X68000専用のMPUアクセラレータXellent30(s)（以下Xellentと略）が東京システムリサーチから発売されて，もう半年ほどになります。このXellent，かなりの数が出ているようです。先月号でも紹介されましたので，読者の皆さんのなかでもすでに購入された方も少なくないと思います。

このXellentでは倍クロックの68EC030と68882が搭載されていますが，それだけではH.A.R.Pの二の舞では？　と思われる方もいらっしゃるかもしれません。68EC030にはプログラムとデータそれぞれ256バイトのキャッシュを積んでいますので，キャッシュONにしておけばそれなりに速くはなります。しかし，キャッシュから溢れた場合は10MHzないしは16MHzのメモリにアクセスせざるをえず，当然速度低下を引き起こします。

そこで，XellentにはローカルSRAMを256Kバイト搭載し，そのSRAMにはノーウェイトでアクセスできるようになっています。しかしこれもセカンドキャッシュとして利用できるわけでもなく，現時点ではXellent30sに付属のloadhigh.rでの使用しか道はありません。loadhigh.rとはE.Watanabe氏作成の実行ファイルのSRAMへのローダなのですが，当然256Kバイト（SRAMにプログラムを常駐させている場合にはそれ以下）を超える大きなプログラムをSRAMに転送することはできません。

また，SRAM上で動作しているプログラムでも，DOS MALLOCによるメモリ確保は無条件でメインメモリから確保されてしまいますし，SRAMはDMACからは見えませんので，DMA転送を1カ所でも行っているプログラムをロードすると，間違いなく暴走してしまいます。「いまどきDMA転送を使ってるプログラムなんてないよ」と思われたあなた，それは甘い考えです。プログラム中では使っていないように見えても，実はファイル転送にはDMAが使われているのです。

したがって，ファイルを扱うプログラムはほぼ全滅といっていいでしょう。これを避けるには，HSCSI（フリーウェア）のようなファイル転送をCPUにやらせるツールを常駐させるという手があります。しかし，これは根本的な解決にはなっていません。SRAMの扱いにくさは相変わらず残っているからです。そこで，今回はプログラム，主にC言語からのSRAMの活用法について考えてみましょう。

図1　メモリ管理ポインタの構造

dc.l	前のメモリ管理ポインタ（前がなければ0）
dc.l	親のメモリ管理ポインタ（親がいなければ0）
dc.l	このメモリブロック+1のアドレス
dc.l	次のメモリ管理ポインタ（次がなければ0）

メモリ確保

loadhigh.rは実行ファイルをSRAMに転送するツールであることは先ほど述べましたが，これを使用するにはあらかじめCONFIG.SYSにXT30DRV.SYSを登録しておく必要があります。こちらはメモリ上のHuman68kにパッチを当てて，SRAMをメインメモリから切り離し，SRAMとメインメモリを個別に管理するようにするためのデバイスドライバです。ただし，Human68k自身がSRAMからメモリを確保できるようになるわけではありません（解放はできますが）。あくまで管理するだけです。ここでloadhigh.rについて考えてみましょう。

loadhigh.rの動作手順としては，

1)　プログラムのサイズを調べる
2)　それ以上のサイズのメモリをSRAMから確保する
3)　プログラムをSRAMに展開する
4)　loadhigh.r自身を解放して展開したプログラムに処理を移す

となっているはずです。

注目すべきは2)の動作です。どうやってSRAMからメモリを確保しているのでしょうか。答えは簡単，自前でやっているのです。Humen68kの内部ワークからメモリ管理ポインタをたどり，自分でメモリブロックを作成しているのです（内部ワークのアドレスはHuman68kのバージョンによって異なりますので，ver.3.02以外では動作しません)。メモリ管理ポインタについてはXC付属のプログラマーズマニュアルで解説されていますが，ここで簡単に説明しておきましょう。

Human68kの動作中は複数のメモリブロックが存在し，それぞれが干渉しないように双方向チェーン構造でリンクされたメモリ管理ポインタ（図1）によって管理されています。このメモリ管理ポインタはHuman68kでメモリを確保した場合には自動的に生成されるもので，実はDOS_MALLOCで確保したメモリの直前にはこの16バイトのメモリ管理ポインタが張り付いているのです。

また，メモリにアロケートされたプログラムについても，先頭にはこのメモリ管理ポインタがあり，その後ろにはプロセス管理ポインタがあります。プロセス管理ポインタについてはここでは触れませんが，図1の親のメモリ管理ポインタとは，自分を確保したプログラムのメモリ管理ポインタを指しています。

これらのメモリ管理ポインタは必ずアドレスの低いほうから高いほうへリンクされていますので，あっちこっちに行ったり来たりすることはありません。また，メモリブロックは16バイト単位で管理されていますので，SIZEバイトのメモリを確保した場合は，実際にはSIZE以上の16の倍数＋メモリ管理ポインタ16バイトの大きさのメモリブロックが生成されることになります。ちなみに標準関数のmalloc()はヒープからメモリを取ってきますので，メモリ管理ポ

インタは生成しません。

XT30DRV.SYSによってパッチを当てられたHuman68kでも，このメモリ管理ポインタの仕組みは変わりません。ただDOS_MALLOCがSRAMの先頭アドレス$BC0000より上を見にいかなくなるだけです。そこで，loadhighのソースなどを参考に，SRAMからメモリを確保する関数（リスト1）を作ってみました。早い話が，メモリ管理ポインタをたどっていき，$BC0000より上に隙間があるかを調べ，あったらメモリ管理ポインタを生成してメモリブロックをどかっと取ればいいわけです。

関数smalloc()は，引数として確保するメモリのバイトサイズを取り，成功した場合は確保したメモリへのポインタを，失敗した（メモリに空きがなかった）場合にはNULLを返します。手抜きのため，最大バイト数を返すことはしません。確保されたメモリは，DOS_MFREEで解放することができます。

また，この関数はXT30DRV.SYSが登録されており，Human68kのバージョンが3.02であることを前提に作られていますが，それらのチェックは行っていませんので，呼び出し側のプログラムでチェックするようにしてください。XT30DRV.SYSの登録のチェックは，"@XT30DRV"というデバイスがオープンできるかどうかで行うことができます。

ついでにsmalloc()の動作を確認するプログラムSmalCheck（リスト2）も作っておきましたので，参考にするとよいでしょう（コンパイル方法はリスト3）。いい忘れていましたが，私はCのライブラリはXCに付属のものを使用していますので，libcを使用する場合には各自で関数名などを直してください。

SmalCheckは，16バイト，16Kバイト，240Kバイトのメモリをsmalloc()関数を使ってSRAMから順次確保します。その都度process.xを呼びますので，確保されたメモリを確認してください。開始アドレスが$BC0000以上のものはSRAMに確保されていることを示します。loadhigh.rを使ってSRAMになにかを常駐させている場合には，おそらく3度目の240Kバイトは確保できないでしょう。

関数単位でのSRAM実行

さて，SRAMからメモリを確保できるようになりましたが，そうはいってもSRAMは256Kバイト，それほど大きなデータを入れられるわけではありません。メモリを確保できるだけでは意味がない，とはいいませんが，あまり嬉しくありません。

たとえばloadhigh.rでSRAMに入りきらないほど大きなプログラムがあったとします。画面周りとか，ユーザーインタフェイスなんかでごてごてしてしまってはいるけれど，実際に「もっと速くなるといいなぁ」という部分は，その核となる数十Kバイトの関数が大部分を占めていたりします。「じゃあ，その関数だけでもSRAMに……」っていうのは自然な発想だったりするわけで，ちょっとそのへんを試してみましょう。

楽観的に考えると，smalloc()でSRAMからメモリを確保して関数を転送し，そこにサブルーチンジャンプする，ってだけのようですが，さにあらず。ちょっと考えなければならない部分があったりします。

リスト1

```
  1: *  smalloc.s        (c)Oh!X Isawo-Kikuchi
  2: *  Xellent30(s)専用ローカルSRAM確保関数
  3: *  Human68k version 3.02専用
  4: *  必ずXT30DRV.SYSを登録しておくこと
  5: * 機能
  6: *  SRAMからメモリを確保する
  7: * 書式
  8: *  void     *smalloc( size );
  9: *  size_t  size:   /* 割り付けるメモリ領域のサイズ */
 10: * 戻り値
 11: *  割り付けた領域を指すポインタを返す。
 12: *  メモリに空きがないときはNULLを返す。
 13: * 備考
 14: *  解放には、DOSコールMFREEを使用できる
 15:
 16:    .include        doscall.mac
 17:
 18:    .xdef   _smalloc
 19:
 20: lstmem      equu    $1c00   * 空きメモリの最終アドレス+1（SRAM含まず）の格納領域
 21: fstmem      equ     $1c04   * 最初のメモリ管理ポインタの格納領域
 22: processmem equ      $13d0a  * 現在のプロセスのメモリ管理ポインタの格納領域
 23:
 24: SRAMtopaddress      equ     $00bc0000
 25: limit_address       equ     $00c00000
 26:
 27:    .offset 0       * メモリ管理ブロック
 28: last:       .ds.l   1           * 一つ前のメモリ管理ポインタ
 29: prcs:       .ds.l   1           * 親プロセスのメモリ管理ポインタ
 30: end:        .ds.l   1           * メモリブロックの終わり+1
 31: next:       .ds.l   1           * 次のメモリ管理ポインタ
 32:
 33:    .text
 34:    .even
 35:
 36: _smalloc:
 37:    move.l  4(sp),d0            * size
 38:    bne     start
 39:    rts
 40: start:
 41:    movem.l d1-d4/a0-a1,-(sp)
 42:
 43:    move.l  d0,d4
 44:    clr.l   -(sp)
 45:    DOS     _SUPER              * スーパーバイザーモードへ
 46:    addq.l  #4,sp
 47:    exg.l   d0,d4
 48:
 49:    add.l   #$10,d0             * メモリ管理ブロックのぶん
 50:    cmpi.l  #limit_address-SRAMtopaddress,d0
 51:    bhi     error
 52:    movea.l fstmem,a0           * 最初のメモリ管理ポインタ
 53:    move.l  lstmem,d2           * 空きメモリの最終アドレス+1
 54: chain_loop:
 55:    move.l  next(a0),d1         * 次のメモリ管理ポインタ
 56:    beq     sram_nouse
 57:    cmp.l   d2,d1
 58:    bcc     reach_sram
 59:    movea.l d1,a0
 60:    bra     chain_loop
 61: reach_sram:
 62:    move.l  #SRAMtopaddress,d2
 63:    move.l  d2,d3
 64:    add.l   d0,d3
 65:    cmp.l   d1,d3               * 隙間に入るか?
 66:    bls     link_mem
 67:    movea.l d1,a0
 68: sram_chain_loop:
 69:    move.l  end(a0),d2
 70:    add.l   #$0000000f,d2       * ブロック整合
 71:    and.l   #$fffffff0,d2
 72:    move.l  next(a0),d1
 73:    beq     end_chain
 74:    move.l  d2,d3
 75:    add.l   d0,d3
 76:    cmp.l   d1,d3               * 隙間に入るか?
 77:    bls     link_mem
 78:    movea.l d1,a0
 79:    bra     sram_chain_loop
 80: sram_nouse:
 81:    movea.l #SRAMtopaddress,a1
 82:    bra     built_mempointer
 83: end_chain:
 84:    move.l  #limit_address,d3
 85:    move.l  d2,d1
 86:    add.l   d0,d1
 87:    cmp.l   d1,d3               * 隙間に入るか?
 88:    bcs     error
 89: link_mem:
 90:    move.l  d2,a1               * 新しいメモリ管理ポインタ
 91:
 92: *  メモリ管理ブロックの生成
 93: *  a0.l    一つ前のメモリ管理ポインタ
 94: *  a1.l    生成するメモリ管理ポインタ
 95: built_mempointer:
 96:    move.l  next(a0),d1         * 次のメモリ管理ポインタ
 97:
 98:    move.w  sr,-(sp)
 99:    or.w    #$0700,sr           * 割り込み禁止
100:
101:    move.l  a1,next(a0)
102:    move.l  a0,last(a1)
103:    move.l  processmem,prcs(a1)
104:    add.l   a1,d0
105:    move.l  d0,end(a1)
106:    move.l  d1,next(a1)
107:    beq     built_end
108:    movea.l d1,a0
109:    move.l  a1,last(a0)
110: built_end:
111:    move.w  (sp)+,sr            * 割り込み許可
112:
113:    move.l  a1,d0
114:    add.l   #$10,d0
115: quit:
116:    tst.l   d4
117:    blt     skip_user
118:    exg.l   d0,d4
119:    move.l  d0,-(sp)
120:    DOS     _SUPER              * ユーザーモードへ
121:    addq.l  #4,sp
122:    move.l  d4,d0
123: skip_user:
124:    movem.l (sp)+,d1-d4/a0-a1
125:    rts
126:
127: error:
128:    moveq.l #0,d0
129:    bra     quit
130:
131:    .end
```

まずは変数の参照。ローカル変数はスタックに取られるからいいとして，関数が移動してしまった場合，その関数内から参照しているグローバル変数はちゃんと見えるのか。もうひとつは分岐。移動した関数内からほかの関数を呼んだとき，正しいアドレスに分岐できるのか。あるいは移動した関数内でのジャンプはどうなるか。この辺は絶対/相対アドレスっていう参照あるいは分岐で回避できるのですが，ここではC言語からの対処法を考えてみます。

●変数の参照

グローバル変数は移動させないわけですから，そのアドレスは動きません。したがって，絶対アドレス参照してやればなんの問題もありません。C言語ではどうかというと，一般的なC言語がどうかは知りませんが，少なくともXCやGCCでは特殊なオプションを付けない限り，グローバル変数は絶対アドレス参照されます。よって，あまり考える必要はありません。

ただし，注意が必要なのは，たとえば，

```
printf( "Xellent" );
```

などとした場合，"Xellent"という文字列は静的な領域に割り付けられるにもかかわらず，関数内からは相対アドレス参照されます。よって，こういった場合には，

```
char    s[] = "Xellent";
        ⋮
printf( s );
```

などとしなければなりません(sはグローバル宣言)。

その他，関数内で宣言された変数であってもstatic宣言された場合には，静的な領域に割り付けられますので，避けたほうが無難です。

●分岐

これも少し考えればわかるとおり，ほかの関数への分岐ならば絶対アドレス分岐，同じ関数内なら相対アドレス分岐してやれば問題なしです。C言語では，こちらも一般的な話かどうかはわかりませんが，まったく上の説明どおりの分岐をしてくれます。よって，こちらも考える必要はほとんどありません。

ただし，相対アドレス分岐は最長±32Kバイトですので，同じ関数内の分岐でも分岐先が遠いと絶対アドレス分岐になるかもしれませんが，普通は問題にならないでしょう。

これらのことからわかるように，実は「ほとんどなにも考えなくていい」のです。もちろん，GCCのオプションで-fall-bsrをつけてコンパイルしたり，最後にCV.Xでリロケータブルや絶対アドレス形式に変換しなければの話ですが。

さて，ここでサンプルプログラムに行きたいところなんですが，もうひとつだけ考えてからにしましょう。

スタック

先ほども少し出てきましたが，スタックとは，引数の受け渡しやサブルーチンの帰りアドレス，あるいはレジスタ保存，また，C言語に限らなければ，プログラマの気分次第でさまざまなワークに使用され，かなり頻繁にアクセスされるものです。したがって，スタックをSRAM上に生成すれば，劇的な高速化とはいかないものの，それなりに満遍なく速くなることが期待できます。

さて，本来はプログラムが起動された時点では，スタックは親プロセスのものが使われます。コマンドラインから起動されたのであればCOMMAND.Xのスタックですね。

しかし，いつまでも親のすねをかじっていたのでは申し訳ないということで，プログラムはまず自分のヒープ内にスタックを作って，そこにスタックポインタを移すのが一般的です。ではC言語の場合はというと，main()関数にくる前処理で自動的にそれを行ってくれています。いったん作ってしまったスタックを無駄にしてしまうのは気がひけますが，実はスタックを速くするというのはほかにも意味があるのです。

スタックとひと言でいってしまいましたが，スタックにはユーザースタックとスーパーバイザスタックがあります。それぞれユーザーモードとスーパーバイザモードで使用されるスタックなのですが，プログラムが動きだした時点ではユーザーモード，すなわちユーザースタックが使用されています。

C言語で作ってくれるスタックもこのユーザースタックで，この時点ではスーパーバイザスタックはまだ親のスーパーバイザスタックを使用しています。「使用っていったって，ユーザーモードだからユーザースタックしか使ってないじゃんか」と思われるかもしれませんが，実はそうではありません。DOSコールやIOCSコールなどの例外処理は，コールされた時点でスーパーバイザモードに移行し，スーパーバイザスタックを使用するのです。これはプログラム中からは認識できない各種割り込みなどにも同じことがいえます。

つまりスーパーバイザスタックをSRAMに載せておくだけで，必然的にそれらが勝手に速くなってくれるわけです。これはちょっと美味しい話だと思いませんか？

スタックを生成するのは別に難しいことではありません。メモリを確保しておいて，スタックのアドレスを示すa7レジスタに確保したメモリの最後のアドレス＋1を入れてやるだけです。なんで最後のアドレスかって？　それはスタックがLIFOバッファで，アドレスの高いほうから低いほうへ積んでいくことになっているからです。説明になってないかもしれませんが，詳しく知りたい人はアセンブラの解説書をどうぞ。

リスト2

```
 1: /* SmalCheck.c      (c)Oh!X Isawo-Kikuchi    */
 2: /* smalloc()関数動作チェックプログラム           */
 3:
 4: #include    <stdlib.h>
 5: #include    <stdio.h>
 6: #include    <process.h>
 7: #include    <doslib.h>
 8:
 9: #define     SIZE1   16
10: #define     SIZE2   16*1024
11: #define     SIZE3   240*1024
12:
13: extern int smalloc( int );
14:
15: void        main()
16: {
17:     int     fileno, adr, i;
18:     int     SIZE[3] = { SIZE1, SIZE2, SIZE3 };
19:
20:     if( VERNUM()!=('68'<<16)+3*256+2 ){       /* バージョンチェック */
21:             printf( "Human68kのバージョンが違います。¥n" );
22:             return;
23:     }
24:     if( (fileno=OPEN( "@XT30DRV", 0 ))<0 ){ /* XT30DRV.SYS登録チェック */
25:             printf( "XT30DRV.SYSが登録されていません。¥n" );
26:             return;
27:     }
28:     CLOSE( fileno );
29:
30:     for( i=0; i<3; i++ ){
31:             printf( "SRAMから %d バイト確保してみます。¥n", SIZE[i] );
32:             adr = smalloc( SIZE[i] );
33:             if( adr==NULL ){
34:                     printf( "確保できませんでした。¥n" );
35:                     break;
36:             }
37:             printf( "確保した領域へのアドレス:%06X¥n", adr );
38:             execlp( "process", "process", NULL );
39:             MFREE( adr );
40:             getch();
41:     }
42: }
```

リスト3

```
GCCの場合
A>gcc smalcheck.c smalloc.s -O doslib.l floatfnc.l

XCの場合
A>cc smalcheck.c smalloc.s /O /Y
```

さて，実際のスタック移動ですが，関数を一発ぽんと呼んでやるだけ，というわけにはいきません。なぜならすでにスタックには戻りアドレスやローカル変数が割り付けられているからです。スタックに積まれているデータを根こそぎコピーしてやれば問題ないとは思うのですが，面倒なのでスタックを移動したあとに指定されたアドレスをサブルーチンコールする仕様にしました（リスト4）。

トンネルをくぐるといつの間にかスタックがSRAMに移り，再びトンネルをくぐって戻ってくると，スタックも元に戻っているといった感じですね。移されるスタックはトンネルをくぐるときに有効だったスタックですが，DOS_SUPERなどでスーパーバイザモードに移行した時点で，ユーザースタックポインタはスーパーバイザスタックポインタにコピーされますので（つまりユーザースタックとスーパーバイザスタックが同じものになる)，ユーザースタックをSRAMに移動させてからスーパーバイザモードに移っても同じことです。

また，モードが違ってもトンネルの先でさらに違うトンネルをくぐることはできないようになっています。スタックのサイズは4Kバイトにしてありますが，足りない場合はSTACKSIZEの値を書き換えてください。たいていは4Kバイトあれば十分でしょう。

また，もし分岐先が戻り値を取るような関数であった場合，たとえばintを戻り値とする関数だった場合には，リスト5の，

extern void　built_stack(void);

を，

extern int　built_stack(int);

と書き換えてやれば，built_stack()関数から戻り値を拾うことができます。

ベンチマーク

さて，いよいよ速度を測定してみましょう。今回ベンチマークプログラムとして作ってみたのは，256×256ドットのグラフィックを128×128に縮小し，だんだん濃度を濃くしながら画面に4×4枚表示したときの経過時間を測定するものです。ちょっと趣味的な方向に走ってしまったうえに，ほかに使い道がないプログラムですが，お許しください。

まずは256×256のデータを作成する必要がありますので，リスト6を実行してください。G-RAM上のハイカラーデータ左上256×256ドットを，bench.datというファイル名でカレントに作成します。

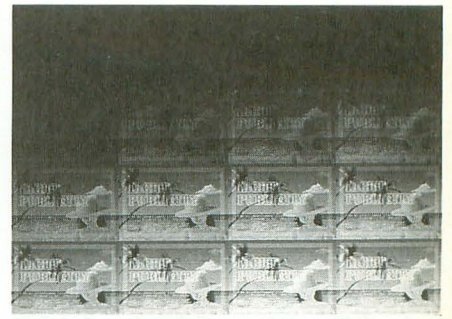
リスト7がベンチマークプログラムですが，このプログラムは大別して以下の5つのパートからできています。

関数	main()	下準備(時間計測外)
	bench()	座標計算・ループ
	pattern()	パターン描画
変数	pat[]	パターンデータ
スタック		

リスト7をBench.cのファイル名でセーブし，リスト8のメイクファイルでコンパイルすると，パートごとにSRAMを使用した以下の5つのプログラムが生成されます

●**Bench1.x**

すべてをメインメモリで，あるいはloadhighですべてをSRAM上で動作

●**Bench2.x**

グラフィックアクセスを計測する

pattern()関数のみをSRAM上で動作

●**Bench3.x**

bench(),pattern()関数をSRAM上で動作

●**Bench4.x**

パターンデータのみをSRAM上で動作

●**Bench5.x**

スタックのみをSRAM上で動作

キャッシュにすっぽり入ってしまっては，SRAMの効力がわかりませんので，pattern()関数でx4()マクロを使ってわざわざループを展開してキャッシュ対策を行っているのですが，4ピクセルを加算するループ内は結局キャッシュに入ってしまっています。まあ，まったくキャッシュに入らないプログラムというのは考えにくいので，これくらいはいいことにしましょう。Bench1をloadhighする場合やBench4は，HSCSIなどでファイル転送をCPUにやらせるようにしておいてください。

ベンチマーク結果を表1に示します。参考までにX68030と68000モードの結果も併記してみました。これによると，SRAMの使用でそこそこ速くなることが確認できるものの，思っていたほどではありませんで

リスト4

```
 1: *  stack.s          (c)Oh!X Isawo-Kikuchi
 2: *  Xellent30(s)専用SRAMスタックフレーム生成関数
 3: * 機能
 4: *  SRAMにスタックフレームを形成し、示されたアドレスに分岐する
 5: * 書式
 6: *  void     built_stack( adr );
 7: *  void     *adr;   /* 分岐アドレス */
 8: * 戻り値
 9: *  Xellent.h 内で記述されたプロトタイプ宣言の戻り値の型を書き替えることで、
10: *  分岐したサブルーチンの戻り値を拾うことが可
11: * 備考
12: *  SRAMが確保できなかったときは、スタックフレームを形成せずに分岐する
13:
14:    .include        doscall.mac
15:
16: STACKSIZE  equ     $1000
17:
18:    .xdef   _built_stack
19:
20:    .xref   _smalloc              * smalloc.s
21:
22:    .text
23:    .even
24:
25: _built_stack:
26:    link    a6,#0
27:    movem.l d7/a0,-(sp)
28:    movea.l 8(a6),a0              * 分岐アドレス
29:
30:    move.l  #STACKSIZE,d0         * スタックサイズ
31:    move.l  d0,-(sp)
32:    bsr     _smalloc              * SRAMからメモリ確保
33:    addq.l  #4,sp
34:    tst.l   d0
35:    beq     jump_adr              * メモリ確保できなかった
36:    move.l  d0,top_stack          * スタックの先頭アドレス
37:    addi.l  #STACKSIZE,d0         * スタックの最終アドレス+1
38:
39:    exg.l   d0,sp                 * スタックを切り替える
40:    move.l  d0,old_stack      * 古いスタックアドレスを保存
41: jump_adr:
42:    jsr     (a0)              * 分岐
43:
44:    move.l  old_stack,d7
45:    beq     quit              * スタックフレームは形成されていない
46:    clr.l   old_stack
47:    move.l  d7,sp             * スタックポインタを戻す
48:    move.l  d0,d7
49:    move.l  top_stack,-(sp)
50:    DOS     _MFREE            * SRAM解放
51:    addq.l  #4,sp
52:    move.l  d7,d0
53: quit:
54:    movem.l (sp)+,d7/a0
55:    unlk    a6
56:    rts
57:
58:    .text
59:    .even
60:
61: top_stack:
62:    .dc.l   0         * SRAMに生成されたスタックの先頭アドレス
63: old_stack:
64:    .dc.l   0         * スタックのアドレス
65:
66:    .end
```

リスト5

```
1: /*      Xellent30(s)専用関数    (c)Oh!X Isawo-Kikuchi   */
2: /*      Human68k version 3.02専用                        */
3: /*      必ずXT30DRV.SYSを登録しておくこと                 */
4:
5: extern void     *smalloc( size_t );     /* smalloc.s */
6: extern void     built_stack( void * );  /* stack.s */
```

した。

というのは，X68030は25MHzであるのに対して，Xellent30は33MHzなので，Bench1をloadhighした場合は絶対にXellent30のほうが速くなると思ったからです。考えられることはただひとつ，どうやらXellent30(の68030モード)はVRAMアクセスに鬼のようなウェイトが入るようです。X68030のVRAMのクロックは12.5MHzですから，実質10MHzか，場合によってはそれ以下になってしまっているようです。

これは，XVIのVRAMが16MHzであることを考えると，VRAMアクセスの比重の高いアニメーションなどのようなプログラムでは，どんなに頑張っても68030モードのほうが遅くなることが予想できます。これは速度比から考えて，Xellent30sでも同様でしょう。ちなみにメインメモリに関しては，これほどひどいウェイトは入らないようです（多少は入るのでしょうが）。

Bench2とBench3を比較してみると，bench()関数も一緒にSRAMに転送したBench3のほうが遅くなっています。最適化の関係くらいしか考えられないので，小さな関数なら下手にSRAMに転送してコンパイラからわかりにくくするよりも，コンパイラの最適化に任せたほうがいいということなのかもしれません。

パターンデータをSRAMに載せたBench4では，ほとんど速くなっていません。のべ2Mバイト分のアクセスですが，データキャッシュに載ってしまっている気配ですので，こんなものでしょうか。それよりもスタックをSRAMに載せたほうが速いという結果になっています。

どうもベンチマークプログラムがあまりよくなかったようですね。すみません。しかし，だいたいの感じはつかめたのではないでしょうか。SRAMを有効に活用すれば，かなりのパフォーマンスアップにつながることは事実ですので。

自由課題

さて，ここまでローカルSRAMの活用ということでやってきましたが，特に有効なのは関数のSRAM転送ではないかと思います。しかし，いったんメインメモリに読み込んでからSRAMに転送していたのでは，メモリの無駄になりますし，なにより美しくありません。それを避ける手段としては，あらかじめSRAMに載せたい関数だけをファイルに落としておいて，それを直接SRAMにロードするということが考えられます。必要な変数や関数のアドレスはすべて引数で渡してやるようにすれば，不可能ではありません。

しかし，ここで注意が必要となるのは，C言語で書いたプログラムでは，プログラマの知らないうちに関数を呼んでいることがある，ということです。たとえば，不動小数点演算をすれば当然のこと，整数演算でもint同士の掛け算などは関数を呼んでいるのです。簡単な関数ならともかく，これをすべて避けるのは容易なことではないでしょう。

そこで，少し重くなりますが，もっと安全な方法を考えてみます。たとえば，SRAMに載せたい関数を含んだ常駐プログラムを作ります。その関数を呼びたいプログラムは，その常駐プログラムを起動し，その際に関数のアドレスと常駐アドレスをもらっておきます。そうすればその関数を自由に呼ぶことができますし，常駐解除は親側でDOS_MFREEしてやればよいので，常駐プログラム側で常駐解除のコードを書く必要もありません。

また，常駐側はいったん常駐してしまえ

リスト6

```
 1: /* ベンチマーク用データ作成プログラム        (c)Oh!X Isawo-Kikuchi    */
 2: /* BenchPic.c                                                        */
 3:
 4: #include    <stdlib.h>
 5: #include    <stdio.h>
 6: #include    <doslib.h>
 7: #include    <iocslib.h>
 8:
 9: void        main()
10: {
11:     FILE    *fp;
12:     unsigned short  *vp = (unsigned short *)0xC00000;
13:     int     i;
14:
15:     fp = fopen( "bench.dat", "wb" );
16:     if( fp==NULL ){
17:             printf( "データをつくれません。¥n" );
18:             return;
19:     }
20:     CRTMOD( 12+0x100 );
21:     SUPER( 0 );
22:     for(i=0;i<256;i++,vp+=512) fwrite( vp,sizeof( unsigned short),256,fp );
23:     fclose( fp );
24: }
```

リスト7

```
 1: /* Xellent30(s)評価用ベンチマーク           (c)Oh!X Isawo-Kikuchi    */
 2: /* Bench.c                                                           */
 3: /* SRAMに転送するパートをマクロで定義する                              */
 4:
 5: #include    <stdlib.h>
 6: #include    <stdio.h>
 7: #include    <time.h>
 8: #include    <process.h>
 9: #include    <doslib.h>
10: #include    <iocslib.h>
11: #include    "Xellent.h"
12:
13: #define     x4(a)   a;a;a;a;
14:
15: void        true_bench( void );
16: void        true_pattern( int, int, unsigned short *, int );
17:
18: unsigned short      *pat;
19:
20: #ifndef     PATDAT
21: unsigned short      patdat[256*256];
22: #endif
23:
24: void        (*bench)();
25: void        (*pattern)();
26:
27: void        true_bench()
28: {
29:     int     i;
30:
31:     for( i=0; i<16; i++ ) pattern( (i%4)*128, (i/4)*128, pat, i );
32: }
33:
34: void        true_pattern( int x0, int y0, unsigned short *p, int n )
35: {
36:     unsigned short  *vp = (unsigned short *)0xC00000;
37:     int     x, y, i, j;
38:     unsigned short  c;
39:     int     r, g, b;
40:
41:     for( y=0; y<128; y++ ){
42:             for( x=0; x<128; ){
43:                     x4(x4(  /* キャッシュ対策 */
44:                             b = r = g = 0;
45:                     /* 4ピクセルを足す(ここはキャッシュに入ってしまう) */
46:                             for( j=0; j<2; j++ ) for( i=0; i<2; i++ ){
47:                                     c = p[(x+x+i)+((y+y+j)<<8)];
48:                                     b += (c>>=1)&0x1F;
49:                                     r += (c>>=5)&0x1F;
50:                                     g += (c>>=5)&0x1F;
51:                             }
52:                             b *= n+1; r *= n+1; g *= n+1;
                                                /* 位置によって濃度を変える */
53:                             b += 32; r += 32; g += 32;/* 四捨五入用のゲタ */
54:                             b >>= 4+2;                  /*最大濃度+4ピクセル分*/
55:                             r >>= 4+2;
56:                             g >>= 4+2;
57:                             vp[(x0+x)+((y0+y)<<9)] = ((((g<<5)|r)<<5)|b)<<1;
58:                             x++;
59:                     ))
60:             }
61:     }
62: }
63:
64: void        main()
```

ばスタックもヒープも必要なくなりますので（コールした親のスタックなりヒープが使用される），常駐終了時に解放してしまえばメモリの圧迫が少なくてすみます。実はこの方法はもうすぐ発売予定のEX-Systemの一部で使われていたりします。興味のある方は試してみるとよいでしょう。

スタックとヒープといえば，ちょっと注意しなければならないことがあります。Cでコンパイルされた実行ファイルは，迷惑なことにデフォルトでスタックとヒープを64Kバイトずつ，計128Kバイト確保してくれます。当然そのほかにコードやデータがあるわけですから，256KバイトのローカルSRAMにはそのままでは実行ファイルがひとつしか入らないことになります。

スタックは先ほども述べたように4Kバイトあれば普通は足りますし，ヒープもmalloc()を使わないのであれば必要ないはずです(最小値は8Kバイトですが)。これらの大きさを変更するには，リスト8のようにコンパイル時にGCCの場合は，-z-stack=SIZE,-z-heap=SIZEオプションで，XCの場合には/Gs,/Ghオプションでサイズを指定します。私はなんであんなにでかくなるのか悩んでしまって，ディスアセンブルまでしてしまったよ，まったくぅ(U氏調)。

表1　ベンチマーク結果

Xellent30s ... SUPER + Xellent30s
Xellent30 ... XVI + Xellent30
Xellent30(s)の68030モードはcache on
単位は10ms

Bench1.x	
X68030(25MHz) cache on	930
X68030(25MHz) cache off	1295
Xellent30s 68000(10MHz) mode	5107
Xellent30s 68030(20MHz) mode	2245
Xellent30 68000(16MHz) mode	2995
Xellent30 68030(33MHz) mode	1320
Bench1.x(loadhigh)	
Xellent30s 68030 mode	1635
Xellent30 68030 mode	975
Bench2.x	
Xellent30s 68030 mode	1840
Xellent30 68030 mode	1098
Bench3.x	
Xellent30s 68030 mode	1882
Xellent30 68030 mode	1102
Bench4.x	
Xellent30s 68030 mode	2214
Xellent30 68030 mode	1306
Bench5.x	
Xellent30s 68030 mode	2130
Xellent30 68030 mode	1258

```
 65: {
 66:     FILE     *fp;
 67:     int      ssp;
 68:     time_t   t;
 69:     short    *p1, *p2, *p3;
 70:
 71: #ifdef      BENCH
 72:     p1 = (short *)true_bench;
 73:     p2 = (short *)true_pattern;
 74:     p3 = (short *)bench = (void *)smalloc( (int)p2-(int)p1 );
 75:     if( p3==NULL ){
 76:             printf( "SRAMが確保できません。¥n" );
 77:             return;
 78:     }
 79:     while( p1<p2 ) *(p3++) = *(p1++);
 80: #else
 81:     bench = true_bench;
 82: #endif
 83:
 84: #ifdef      PATTERN
 85:     p1 = (short *)true_pattern;
 86:     p2 = (short *)main;
 87:     p3 = (short *)pattern = (void *)smalloc( (int)p2-(int)p1 );
 88:     if( p3==NULL ){
 89:             printf( "SRAMが確保できません。¥n" );
 90:             return;
 91:     }
 92:     while( p1<p2 ) *(p3++) = *(p1++);
 93: #else
 94:     pattern = true_pattern;
 95: #endif
 96:
 97: #ifdef      PATDAT
 98:     pat = (unsigned short *)smalloc( 256*256*sizeof( unsigned short ) );
 99:     if( pat==NULL ){
100:             printf( "SRAMが確保できません。¥n" );
101:             return;
102:     }
103: #else
104:     pat = patdat;
105: #endif
106:
107:     fp = fopen( "bench.dat", "rb" );
108:     if( fp==NULL ){
109:             printf( "bench.dat がありません。¥n" );
110:             return;
111:     }
112:     fread( pat, 2, 256*256, fp );
113:     fclose( fp );
114:
115:     CRTMOD( 12 );
116:     G_CLR_ON();
117:     OS_CUROF();
118:     ssp = SUPER( 0 );
119:     t = clock();
120: #ifdef      STACK
121:     built_stack( bench );
122: #else
123:     bench();
124: #endif
125:     t = clock()-t;
126:     SUPER( ssp );
127:     CRTMOD( 16 );
128:     OS_CURON();
129:     printf( "¥n所要時間:%dx10ms¥n", t );
130:
131: #ifdef      BENCH
132:     MFREE( bench );
133: #endif
134: #ifdef      PATTERN
135:     MFREE( pattern );
136: #endif
137: #ifdef      PATDAT
138:     MFREE( pat );
139: #endif
140: }
```

リスト8

```
 1: # Xellent30(s)評価用ベンチマークメイクファイル       (c)Oh!X Isawo-Kikuchi
 2:
 3: CC = GCC
 4: CFLAGS = -Wall -O -fstrength-reduce -fomit-frame-pointer -fno-defer-pop -z-
heap=8192 -z-stack=4096
 5: MARIKO = ABD
 6: GCC_OPTION = FMLOE+
 7: GCC_AS = HAS
 8: GCC_LINK = HLK
 9:
10: AS = HAS
11: AFLAGS = /u /w
12:
13: LINK = HLK -l -o
14: LIBS = CLIB.L GNULIB.L BASLIB.L IOCSLIB.L DOSLIB.L FLOATFNC.L
15:
16: all :      Benchpic.x Bench1.x Bench2.x Bench3.x Bench4.x Bench5.x
17:
18: %.o : %.s
19:     $(AS) $(AFLAGS) $<
20:
21: Benchpic.x : Benchpic.c
22:     $(CC) $^ $(CFLAGS) $(LIBS)
23:
24: # すべてメインメモリ、あるいはloadhighでスタック以外をSRAM上で動作
25: Bench1.x : Bench.c
26:     $(CC) -o $@ $^ $(CFLAGS) $(LIBS)
27:
28: # pattern()関数だけをSRAM上で動作
29: Bench2.x : Bench.c smalloc.o
30:     $(CC) -o $@ $^ $(CFLAGS) -DPATTERN $(LIBS)
31:
32: # bench()とpattern()関数をSRAM上で動作
33: Bench3.x : Bench.c smalloc.o
34:     $(CC) -o $@ $^ $(CFLAGS) -DBENCH -DPATTERN $(LIBS)
35:
36: # パターンデータだけをSRAM上で動作
37: Bench4.x : Bench.c smalloc.o
38:     $(CC) -o $@ $^ $(CFLAGS) -DPATDAT $(LIBS)
39:
40: # スタックだけをSRAM上、あるいはloadhighですべてをSRAM上で動作
41: Bench5.x : Bench.c stack.o smalloc.o
42:     $(CC) -o $@ $^ $(CFLAGS) -DSTACK $(LIBS)
```

WE WANT YOU!

Oh!Xは，読者の皆さん１人ひとりの力が作り上げていく雑誌です。あなたも誌面作りに協力してくれませんか？

投稿大募集

Oh!Xでは読者の皆さんによる投稿作品を常時募集しています。

未発表の作品であれば，グラフィック，音楽，システムプログラム，ツール，ゲーム，ハードウェアなどジャンルを問いません。機種についても特に限定はしませんが，雑誌の性格上扱いにくい場合もあります。

誌面に載りきらない大きなアプリケーションなどはディスクメディアを使って配布することが考えられます。その形態のひとつはご存じ付録ディスク，そしてもうひとつは別冊形式によるものです（発売中の「Z-MUSICシステムver.2.0」に続き，今後もいくつかのOh!X BOOKSシリーズが予定されています）。

また，「こんなものを作ってみました」といったものでもかまいません。気軽に作品を送ってみませんか。

投稿募集要項

1）お送りいただくプログラムには，住所，氏名，年齢，職業，連絡先電話番号，機種名，使用言語，動作に必要な周辺機器，パソコン歴などを明記のうえ，封書の宛先の最後には「Oh!X LIVE」「全機種共通システム」「投稿ゲームプログラム」など，プログラムの内容を明確にご記入ください。

2）投稿されるプログラムには詳しい内容を記入した原稿を同封してください。ディスクの中にドキュメントファイルの形式でのみ記述している方がいますが，郵送時の事故などでメディアが破壊されることもありますので，必ず文書を添えるようにしてください。変数表，メモリマップ，参考文献などの情報があればなお結構です。また，掲載に際しては，プログラムやデータ原稿に対して加筆修正をさせていただくことがあります。

3）お送りいただくプログラムは事故防止のため最低２回はセーブしておいてください。基本的に原稿などの返送はいたしませんので，あらかじめご了承ください。

4）ハード製作関係の投稿については，最初は内容のわかる原稿のみお送りいただければ結構です。その後，当方で製作物が必要だと判断した場合には改めてご連絡いたします。

5）作品の採用については，掲載号が決定した時点で当方より連絡いたします。特にツールやハード関係などの作品は特集内容などを考慮したうえで採用決定されますので，結果を連絡するまで時間がかかる場合があります。

6）投稿いただいたプログラムにバグなどが発見された場合は，新しいプログラムの入ったメディアと一緒に文書にてご連絡ください。

7）掲載されたプログラムに対しては当社規定の原稿料をお支払いいたします。また，投稿されたプログラムの著作権などはすべて制作者に保留されますが，いわゆる「フリーソフトなどとしてネットにアップする」ことなどを希望される場合には，必ず事前に編集部までご連絡ください。なお，一般的モラルとして，他誌との二重投稿，または他誌に掲載されたプログラムの移植などは固くお断りいたします。

その他，不明な点は編集部までお問い合わせください。

Oh!X編集部　☎03(5642)8122

協力スタッフ募集

Oh!Xでは誌面作りに参加していただく協力スタッフを募集しています。

スタッフとして活動する熱意があり，東京近郊にお住まいの方でソフトバンクに来社可能な方。時間的束縛は特にありませんが，ある程度時間に余裕がある方に限ります。基本的に学生を対象にしていますが，時間的余裕と余力が十分にあれば社会人も可とします。ただし，18歳未満の学生および浪人生の方については採用予定はありません。

応募要項ですが，ライター希望の方はOh!X誌面１ページ分相当（2500字程度）の自由論文に自己紹介文を添えて「Oh!Xスタッフ希望」係までお送りください。

また，文章力には自信がないけどプログラムなら……という方でも技術スタッフとして参加していただく場合があります。こちらを希望の方は，自由論文の代わりにこれまでに制作した自作プログラムとその解説などを一緒に応募してください。

書類選考後，採用の方にはこちらからご連絡いたします。

すべての読者へのお願い

いまはまだ何もできないけれど，いつかは……と思っているアナタにも，いますぐできるいちばん重要なことがあります。アンケートハガキへのご協力です。Oh!Xの誌面の方向性は，このアンケートで寄せられた読者のご意見をもとに決定されています。

皆さんからの熱いメッセージをお待ちしています。

そして，宛先

〒103　東京都中央区日本橋浜町3-42-3
ソフトバンク株式会社
Oh!X編集部　○○○○係

イラスト投稿の規定

サイズはハガキ大（A6判）からB5判くらいまでを目安としますが，取り扱いの手間や現実的な問題としてハガキ大を一応の標準とします。いずれにせよ，掲載時にはかなり縮小されることを考慮して描いてください。

一応の推奨形式は以下のとおりです。

1）ハガキ大のケント紙で郵送

ハガキでも結構ですが，たまに裏面にも消し印が押される危険があります。

2）黒一色（薄ズミ不可）

墨汁は汚れの原因になることがあります。製図用インクがおすすめです。原稿は縮小されますのでスクリーントーンの80，90番台（レトラセットの場合）や色の濃すぎるものなどについての再現は保証しかねます。また，残念ながら，カラー原稿はごくたまにしか掲載されません。

内容に関して特に規制はありませんが，季節ものについては，掲載が予想される時期を考慮して早めに送ったほうが有利になることがあります（年賀状は例外）。

皆さんの力作をお待ちしております。

(で)のショートプロぱーてぃ――その70

戦うっていってもねえ

Komura Satoshi 古村　聡

対戦大砲ゲームにロクロシュミレーター，そしてツールと今月もショートプロは，バリエーション豊かに作品を紹介します。そうそう，今月から始った，没作品にも愛の手を差しのべるコーナー「今月のもう一歩でした」もよろしくね。

illustration:T.Takahashi

この号が出るころにはちったぁ落ち着いた世の中になってるといいんですが，現在，テレビを見ても新聞を見てもサリン，異臭騒ぎ，宗教団体幹部の刺殺，小包爆弾と暗い話題ばっかりで思わず暗い話題を書きそうになってしまうのであります。だって，本当に明るい話題ってないんだものなー，周りも自分も。当然かもしれないけど，やっぱり，戦いだ戦争だなんてのはゲームとかマンガとか空想の中だけなのが一番だよな，と思うのであります。はい。だれか明るい話題を提供してくれないもんかしら。

あ,そういえば,私,最近AmericaOnLineというネットにアクセスしてるんです。その名のとおり，アメリカにあるネットでインターネット経由でアクセスしてるんですが，こいつが面白い。専用のグラフィックインタフェイスをもった通信ソフトでアクセスするんですが，各ボードごとにグラフィカルなセレクトボタンが表示されたり，文書といっしょに写真が表示されたりバシバシグラフィックを多用するパソ通でね。重いけど面白いんですわ，これが。

これって昔から，誰もが考えていたアイデアなんでしょうけど，最近は14400bpsとか28800bpsとか速いモデムが出てきたおかげで,本当にまともな(まだちょっと重いけど)ものができるようになったんですね。速さは力って本当だったんだな，としみじみ考えてしまいます。

あ,「インターネット」「グラフィカルなユーザーインタフェイス」でわかるように残念ながらX68000ではアクセスできません。さっさとシャープさんも新機種出して，戦線復帰できるようにしてほしいもんです。お願いします……うーむ，結局暗い話題になってしまったではないか。

かけひきで勝負だ！

さて，せっかくですから，1本目にはの～てんきな戦争ごっこゲームに登場していただきましょう。でも，本当に面白いんだぞ，このゲームは。進戸さん作のARTILLERY.BASです。どうぞっ。

ARTILLERY.BAS for X680x0
(X-BASIC)
兵庫県　進戸健太郎

このプログラムは2人対戦用の空気抵抗の要素を加えた砲撃ゲームです。

昔々，空気のあるとある惑星に青国と赤国がありました。両国は仲が悪く，常にいがみ合っていました。また両国は，国家のシンボルとして巨大な大砲を持っていました。互いに相手の大砲に脅威を感じていた両国は兵器削減条約を締結するための会議を開きましたが，交渉は決裂しました。そこで両国は，互いに宣戦布告を行い，いままでろくに使ったこともない大砲を国境付近に集結させたのでありますが……。ということで，BASIC.CNFに，

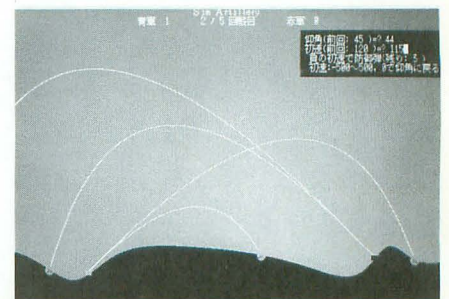
FUNC＝MUSICZ

とMUSICZ.FNCを登録して，リスト1をBASICから打ち込みましょう。で，砲台の発砲音と爆発音の2つのPCMファイルを用意して,自分の環境にあうように130行を書き換えて，間違いがないことを確認してからファイルを保存して……どうだ，一気に読んだら息が切れただろう(なにをやっているのだ)。えーっと，RUN！

起動するとまずタイトルが出るので，なにかキーを押してください。

ゲームは朝から夜にかけての5回戦対戦勝負。それぞれの回の中で，お互いに砲弾を1発ずつ撃ちあって，先に相手に当てたほうが1回勝ちになります。攻撃ターンになると前回の自分の撃った角度と速度が表示されるので，それを参考に角度(°)と初速(m/s)を入力してください。仰角は相手方向水平が0度です。初速は500m/sが最高となっていて，初速を負の値にすると防御弾を発射できます。また，初速入力時に仰角を変更したくなった場合は，初速に0かなにも入力せずにRETURNキーのみを押せば仰角設定に戻れます。

初速の入力が済むと砲弾は弾道を描いて飛んでいきます。このとき，砲弾は空気抵抗と風の影響によって放物曲線とは違う飛び方をします。風の影響はかなりあるのですが，風の方向，強さはわかりません(砲弾を撃ってるとなんとなくわかりますけどね)。

5回戦が終了すると同時に，戦争終結の表示が出て，停戦協定の内容が表示されます。ただし，賠償金以外の表示はありません。相手との得点差によって，賠償金の額が違うので3回戦で3―0になっても「もういーや」などと投げてはあかんのです(実際こんなことといって選手生命投げちゃった，現野球チームのカントクとかいますけど)。結果を表示しているときに，ESCキー以外を押せば，停戦協定を破棄して再び戦争が

ARTILLERY.BAS

始まります。

うひょ〜，燃えます燃えます。最近２人対戦ゲームの投稿が多いんですが（実は，「そろそろコンピュータが相手してくれ」なんて贅沢なこと思ったりしたんですけど），その中でも群を抜くファイアー度だと思います（わざわざ英語でいわんでもよろしい）。相手の着弾が徐々に近づいてくるスリルもさることながら，「やばい！」と思ったら弾幕張って防御できるってのはなかなかいいアイデアですよね。しかもこの弾幕，１回戦に３回までしか使えないうえに，自分の砲撃も妨害されてしまうという条件までついていたりして，かけひきをよりいっそう面白くさせてくれます。対戦ゲームのかけひきはやっぱり，こうでなくちゃぁいけません。

あ，そうそう，この大砲の砲身はあんまり速い砲弾を撃つと少しずつ傷んできて，最後には暴発してしまいます。このへんもかけひきの材料でありますね。

それから，このプログラムはMUSICZのＣコンパイラ用ライブラリがあれば，そのままBASICコンパイラでコンパイルして遊べます。コンパイル時に自動で速度調整されるようにできているので，速くなりすぎてゲームにならないことはありません。GCCでコンパイルする場合は，最適化スイッチ（−O）はつけないでくださいね。

芸術は破裂だ……はぅ

では続いて今月２本目のプログラムは……うーむ，芸術は爆発だ！　バーチャルロクロプログラム，ROKURO.BASであります。どうぞっ！

ROKURO.BAS for X680x0
（X-BASIC，要EXEC.FNC）
岩手県　佐々木崇

ロクロってわかりますよね。漢字で書くと轆轤（ASK……出ないのね，やはし）。そうそう，陶器の器なんかの形を作るのに使う粘土をくるくる回す台です。このバーチャルロクロプログラムでは，X68000上でロクロを再現して，コンピュータ上で造形を行うプログラムです。

ROKURO.BAS

このプログラムはX-BASIC用のプログラムです。1994年10月号「もみじ狩りPRO-68Kディスク」に付属のEXEC.FNCが必要になります。BASICのディレクトリにEXEC.FNCをCOPYして，BASIC.CNFというファイルに，

FUNC＝EXEC

という１行を追加してください。続いて，BASICを立ち上げてからリストを打ち込んでください。で，間違いなく打ち込めたらセーブして，RUN。

画面には何本かの楕円が描かれています。これがロクロ上で回っている粘土です。で，心を落ち着かせてからキーボードをBREAKキーがあるほうを上にして，縦にかまえてくださいまし。キー操作は[ESC][1][2][3]……[-][^][＼][BS]，つまりBSキーからESCキーまでの列を使います。いくつかキーを押してみましょう。すると押したキーに対応する部分が桃色で表示され，へこみます。その上下は逆にふくらみます。

次にCTRLキーを押しながらいくつかキーを押してみてください。すると粘土が押し上げられるようになります。SHIFTキーとCTRLキー両方を押すと内部に手をまわしたことになり，そこから押し広げるという動作になります。これらの操作を組み合せて，壺や椀を作り上げてください。

操作方法の基本は下のほうから徐々にキーボードを撫で上げること。そうしないと，高低がわからなくなってしまいますし，形が整えにくいんですね，これが。あー，チミチミ，ワイヤーの順番を数えてキーボードを触らんように。それって邪道です。や〜，やっぱ芸術家たるもの感覚でもって覚えなきゃねー，はっはっは。

さて，形を整え終わったら，焼き上げです。CLRキーを押すと焼き上げに入ります。機種によっては，焼き上がるのに数分，時間がかかります。

SHIFTキーとCTRLキーを押すと焼き上がった陶器を表示して終了です。さて，無事陶器はできたでしょうか？

……できねい。うみゅ〜，壺ができんぞ壺がぁ！　灰皿はいくらでも作れるんだがな〜（や，適当にこねこねしてるとなんとなくいびつな形の皿はできる，ってーだけの話なんですけどね）。適当にやるぶんには楽しいですけど，思いどおりの形を作ろうとするととっても難しいものですね，これって。本物の陶芸もそういうもんなんでしょうけど。さっきなんか「芸術は爆発だぁ！」というような斬新なデザインの灰皿（笑）を作っていたら，本当に爆発したのかプログラムが陶器の色を塗り漏らしてしまいました。あ゛〜，こんなとこまでバーチャルなのね（実際，本物の陶芸でも無理な形を焼こうとすると，焼いてる最中に割れるそうな。うみゅ〜）。ちなみに焼き始めると，焼き上がるまで，どんな色に焼けるかわからんので，それも楽しみなんですよね。「うむ，なかなかいい色が出たわい」なんてすっかり気分は山にこもる陶芸家でざんす（って壺の作れない陶芸家がいるのか？）。

そうそう，この作者の佐々木さんによると，このようにSHIFTキーやCTRLキーを多用すると，ファンクションキーにのみSHIFTキー，CTRLキーがロックされることがあるんだそうです（F2キーを押した途端，ディスクがイジェクトされたりするんだそうな）。うーむ，BASICインタプリタもコンパイラのライブラリもプログラムを終了時にはキーバッファをクリアさせてるはずなんですけどね，不思議です。だもんで，このプログラムではSHIFTキー，CTRLキーを押させて終了しているのだそうだけど……うーむ，ここのキー操作だけはちょっと気になっちゃいますよね。なんとか解決策はないもんでしょうかね。うむむむむ。

ファンファーレがパンパカパン

最後はOh!Xスタッフからの投稿（？）です。このプログラムは実用プログラムで，Human68k ver.3.0以上専用のコンフィグレーションセレクタです。どうぞ！

BDSELECT.R for X680x0
（要Human68k ver.3.0以上）
埼玉県　江川乃誉司

このプログラムはCONFIG.SYSやAUTOEXEC.BATを含むディレクトリを起動時に選択し起動できるようにする，起

動ディレクトリセレクタです。おまけ機能としてPCM再生機能も搭載されています。

まず，リストをエディタから打ち込んで保存，それからアセンブル，リンク，Rファイルにコンバートすると BDSELECT.R ができます。できましたか？

無事にできたら使い方を説明しましょう。このプログラムはCONFIG.SYS の「EXCONFIG=」に指定して利用します。したがってHuman68k ver.3.0での使用が前提となります。

EXCONFIG=\SYS\BDSELECT.R\SX\GRAPHIC \MUSIC \WP \ETC

と，EXCONFIGの最初のパラメータにこのプログラムを指定して，次に飛び先のディレクトリを指定してください。このようにEXCONFIGのパラメータはディレクトリ５つぶんまで設定できます。それぞれ順にXF1～XF5のキーに対応していて起動時にそれぞれのキーを押すことで，そのディレクトリに移動し起動されます。押されていなかった場合にはディレクトリ移動は行われません。

で，キーを押しているとディレクトリが移動されるわけですから，移動先のディレクトリにAUTOEXEC.BATやCONFIG.SYSがあれば当然それを実行します。つまり，移動先のディレクトリが，ルートディレクトリのAUTOEXECやCONFIG.SYSが移動するディレクトのものと違っていれば，違った環境変数や違うドライバを設定できるのです(逆をいえば，飛び先に設定されたそれぞれのディレクトリにAUTOEXEC.BATとCONFIG.SYSを書かなくてはならない，ということでもあるんですけど)。

そうそう，起動ディレクトリに FANFARE.PCM というファイル名でPCMファイルを置いておけば，起動時に(CONFIG.SYSやAUTOEXEC.BATを実行しながら)再生してくれます。まさしくファンファーレですね。

うーむ，なかなか使い勝手のよさそうなプログラムですね。このプログラムの一番いいところは，作者の江川乃君もいっているんだけど「SRAM常駐型でない」ってことですね。ほかのCONFIGセレクタや電源ONでPCMを鳴らしてくれるプログラムはあったんですけど，たいていSRAMにプログラムもデータも常駐させるタイプだったんですよね。だもんで，どんなにがんばってもPCMデータが32Kバイトまでしかいかなかったのですね。だけど，このBDSELECTであれば「ずっと君色思い～」オープニング全部とか，プリティサミーの歌全部だろうが入るんです。ふふふのふ(いったい何Mバイトハードディスクを無駄使いすれば気がすむのだろう)。

や，いいですねいいですね～。

ちなみに作者の江川乃君は，ズバリ「起動時に『バーチャレーシング』のスタートBGMを通常起動は初級コースでぇー，SXは中級コース，上級コースは……みたいなことがやりたかっただけです」だそーで。うんうん，そうでしょうそうでしょう，うんうん。や～，投稿ディスクも入れたとたんにBGMが流れてきて凝ってましたもんね。作者の愛がひしひしと伝わってきます。

あ～，そうそう，まったくもって個人的な要望なんでありますが，XF1で起動されるシューティングゲームが爽快感あってひっじょうによござんしたしたよ。あれ，なんとか全部で300行ぐらいまで縮めて，ショートプロに投稿してみませんか？

ってことで今月はおしまい。それではまた来月っ！

リスト1　ARTILLERY.BAS

```
 10 /*
 20 /*      Virtua Artillery    by 進戸健太郎 1994-1995
 30 /*
 40 screen 2,0,1,1:console 0,32,0:wipe()
 50 float ex1,ey1,ex2,ey2,ax1,ax2,ev1,ev2,vw,bl=400
 60 float v,s,bv1,bs1,bv2,bs2,er1,er2,erm=500
 70 float gbf(1280)
 80 int i,j,a,b,ply,p1,p2,rst,h,dc1=3,dc2=3 /* plyは +1 or -1
 90 int tr
100 str inp
110 dim str rank(4)=("１０円","","５０円","","１００円")
120 dim str nat(1)=("青国","赤国")
130 m_pcmset(0,"gun3.pcm"):m_pcmset(1,"bomb.pcm")
140 tr=(1=1)
150 SetPalet(0)
160 Title()
170 repeat
180   for j=1 to 5
190     ShowResult():fill(0,0,767,511,1):SetPalet(j):dc1=3:dc2=3
200     repeat
210       ex1=rnd()*384+64
220       ey1=rnd()*512-(612-bl)
230       ev1=rnd()*128
240       ex2=rnd()*384+320
250       ey2=rnd()*512-(612-bl)
260       ev2=rnd()*128
270       ax1=rnd()*192+16
280       ax2=rnd()*192+560
290       ay1=rnd()*256-(512-bl)
300       ay2=rnd()*256-(512-bl)
310       bv1=0:bs1=0:bv2=0:bs2=0
320     until ((bl-511<CalAlt(ax1)) and (CalAlt(ax1)<bl) and (bl-511
<CalAlt(ax2)) and (CalAlt(ax2)<bl))
330     DrawField()
340     DrawArtl()
350     repeat
360       rst=Shot(ply)
370       if (rst) then (
380         locate 43,10:print nat((ply=-1)*tr)+"の勝ち！"
390         repeat:until inkey$(0)<>"" or j=5 )
400       ply=-ply:v=0:s=0
410     until (rst)
420     p1=p1+(ply=-1)*tr:p2=p2+(ply=1)*tr
430     er1=er1*(ply=-1)*tr:er2=er2*(ply=1)*tr
440     ShowResult()
450   next
460   fill(256,128,512,384,(3*(p1>p2)+4*(p1<p2))*tr)
470   locate 42,9 :print "＜戦争終結＞"
480   locate 33,12:print "本日";time$;"をもって、停戦する。
490   locate 38,16:print "なお勝利国である"+nat((p1<p2)*tr)+"が、"
500   locate 36,18
510   print nat((p1>p2)*tr)+"から賠償金"+rank(abs(p1-p2)-1)+"を"
520   locate 36,20:print "支払われることで合意した。"
530   locate 32,23:print "[ENTER]停戦協定破棄[ESC]中立宣言"
540   p1=0:p2=0:er1=0:er2=0
550 repeat:inp=inkey$(0):until inp<>""
560 until inp=chr$(27)
570 screen 2,0,1,1
580 end
590 func Shot(sd)
600    h=3+24*((sd=1)*((bl-CalAlt(ax1))<400)+(sd=-1)*((bl-CalAlt(ax2
))<400))
610   DrawArtl()
620   ShowResult()
630   BkCol(sd)
640   repeat
650     vw=rnd()*2-1:ShowResult()
660     locate 2+64*(sd=-1)*tr,h+3:print"相手方向水平が０°"
670     locate 1+64*(sd=-1)*tr,h
680     print"仰角(前回:";(bs1*(sd=1)+bs2*(sd=-1))*tr;:input")=";s
690   repeat
700     a=(dc1*(sd=1)+dc2*(sd=-1))*tr:v=0:if a<>0 then (
710     locate 2+64*(sd=-1)*tr,h+2
720     print"負の初速で防御弾(残り:";a;")":)
730     locate 2+64*(sd=-1)*tr,h+3
740     print"初速:"+itoa(1-501*(a<>0)*tr)+"～500, 0で仰角に戻る"
```

```
 750     locate l+64*(sd=-1)*tr,h+l
 760     print space$(32)
 770     locate l+64*(sd=-1)*tr,h+l
 780     print"初速(前回:";(bv1*(sd=1)+bv2*(sd=-1))*tr;:input")=";v
 790   until ((a<>0 and -500<=v) or (a<1 and 0<=v)) and v<=500
 800   until v<>0
 810   dc1=dc1-(sd=1)*(v<0):dc2=dc2-(sd=-1)*(v<0)
 820   if sd=1  then { bv1=v:bs1=s:er1=er1+(v-150)*(v>150)*tr+2
 830                   v=v-er1/100:s=s+(rnd()-0.5#)*er1/150 }
 840   if sd=-1 then { bv2=v:bs2=s:er2=er2+(v-150)*(v>150)*tr+2
 850                   v=v-er2/100:s=s+(rnd()-0.5#)*er2/150 }
 860   BkCol(sd):ShowResult()
 870   if (er1>erm and sd=1) or (er2>erm and sd=-1) then {
 880      v=0 :locate 44,8:print"砲身破裂!"
 890   }
 900   m_pcmplay(0,1+(sd=-1)*tr,4)
 910   return(FlyingProcess(sd,v,s,((sd=1)*ax1+(sd=-1)*ax2)*tr,5+CalA
lt(((sd=1)*ax1+(sd=-1)*ax2)*tr),vw))
 920 endfunc
 930 func DrawField()
 940   for a=0 to 767
 950     line(a,bl-CalAlt(a),a,511,2)
 960   next
 970 endfunc
 980 func float CalAlt(a)
 990   return(ey1/(pow((ex1-a)/ev1,2)+1)+ey2/(pow((ex2-a)/ev2,2)+1)+a
y1/(pow((ax1-a)/128,2)+1)+ay2/(pow((ax2-a)/128,2)+1))
 1000 endfunc
 1010 func DrawArtl()
 1020   fill(ax1-6,bl-CalAlt(ax1)-7,ax1+6,bl-CalAlt(ax1),3)
 1030   fill(ax2-6,bl-CalAlt(ax2)-7,ax2+6,bl-CalAlt(ax2),4)
 1040 endfunc
 1050 func FlyingProcess(sd,fv;float,fs;float,fx;float,fy;float,fwx;fl
oat)
 1060   float fvx,fvy,fax,fay,g=9.8#,dt,kpm=0.12#:int pcl,hf
 1070   dt=((tr=-1)*0.05#+(tr=1)*0.01#)*tr
 1080   if fv=0 then sd=-sd : ply=-ply
 1090   fvx=abs(fv)*cos(fs*pi()/180)*sd
 1100   fvy=abs(fv)*sin(fs*pi()/180)
 1110   repeat
 1120     fvx=fvx-kpm*(fvx-fwx)*dt
 1130     fvy=fvy-(g+kpm*fvy)*dt
 1140      fx=fx+fvx*dt
 1150      fy=fy+fvy*dt
 1160     pcl=point(fx,bl-fy)
 1170     pset(fx,bl-fy,5)
 1180   until pcl=2 or pcl=3+(sd=1)*tr or pcl=14 or fx>767 or fx<0 or
bl-fy>511 or (fv<0 and fvy*fvy<1)
 1190   hf=(pcl=3+(sd=1)*tr)*tr
 1200   m_pcmplay(1,3,4-2*hf)
 1210   for i=0 to 8+56*hf:circle(fx,bl-fy,i,6):next
 1220   for i=0 to 8+56*hf+4*(fv<0)*tr
 1230   circle(fx,bl-fy,i,1+(fv<0)*tr*13):next
 1240   return(((pcl=4) and (sd=1)) or ((pcl=3) and (sd=-1)))
 1250   fv=0:fs=0:ft=0:fx=0:fy=0:fvx=0:fvy=0:fax=0:fay=0
 1260 endfunc
 1270 func SetPalet(j)
 1280   /*             0    1空    2山    3左    4右  5弾道
 1290  dim pal(5,5)={20,    30,10280,20530,30780,51580,
 1300               +0,20582,20801,    62, 3714,38858,
 1310               +0,27442,29002,    62, 3714,52850,
 1320               +0,41662,33098,    62, 3714,65535,
 1330               +0,31094,29322,    62, 3714,57334,
 1340               +0,21092,20480,    62, 3714,51176}
 1350   for i=0 to 5
 1360     palet(i,pal(j,i))
 1370   next
 1380   palet(6,38859)
 1390 endfunc
 1400 func BkCol(sd)
 1410   int gx,gy
 1420   gx=(sd=-1)*tr*512
 1430   gy=16*h-8
 1440   if point(gx,gy)<>3+(sd=-1)*tr then {
 1450   get(gx,gy,gx+255,gy+79,gbf)
 1460   fill(gx,gy,gx+255,gy+79,3+(sd=-1)*tr) :} else {
 1470   put(gx,gy,gx+255,gy+79,gbf) :}
 1480 endfunc
 1490 func ShowResult()
 1500   cls
 1510   locate 40,0 :print "Sim Artillery"
 1520   locate 28,1
 1530   print "青軍 ";p1;"      ";j;"/ 5 回戦目          赤軍 ";p2
 1540 endfunc
 1550 func Title()
 1560   for i=0 to 5
 1570     circle(383,255,(i+1)*32,i)
 1580     symbol(210-i*2,200,"Sim Artillery",2,3,2,i,0)
 1590   next
 1600   locate 35,26:print"[RETURN] Start     [ESC] Quit"
 1610   repeat:b=b+1:b=b mod 65535:inp=inkey$(0)
 1620   until inp<>"":if inp=chr$(27) then screen 2,0,1,1:end
 1630   ply=pow(-1,b)
 1640   m_pcmplay(0,3,4)
 1650   for i=0 to 5
 1660     palet(5-i,20):for j=0 to 1000+(tr=1)*9000:next
 1670   next
 1680   m_pcmplay(1,3,4)
 1690 endfunc
```

リスト2 ROKURO.BAS

```
 10 /* ろくろ。                       佐々木崇
 20 int red,grn,blu
 30 dim str h(3)={ "←","↓","↑","→" }
 40 dim y(17),yy(17),r(17),rr(17),p(17),pp(17)
 50 int rrr,yyy,k,sf,l,ll,lll
 60 dim sr(17),er(17),cl(17)
 70 dim c(1) :c(0)=rgb(22,22,22) :c(1)=rgb(31,22,22)
 80 float fr
 90 dim s(17)={6,5,4,3,2,1,1,0,0,0,1,1,2,2,3,4,5,6}
100 dim _bitsns(2)={ &H22127004,&H4E4F2200,&H70004E75 }
110 for l=0 to 17
120   y(l)=480-l*6
130   r(l)=120-(((15-l)*(15-l)) shr 2)
140   sr(l)=180+l*10
150   er(l)=sr(l)+10
160 next
170 y(16)=100
180 screen 1,3,1,1:console 0,31,0
190 print "バーチャSIMろくろPRO-68K"
200 repeat
210   py=0
220   for l=1 to 15
230     k=exec(_bitsns,0) or (exec(_bitsns,1) shl 8)
240     p(l)=(k shr (16-l)) and 1
250     sf=(exec(_bitsns,14) and 3)
260     locate 60,1 :print h(sf)
270     if p(l)=1 then {
280       sf=(exec(_bitsns,14) and 3)
290       py=py+(sf and 1)-(sf shr 1)
300       sf=1+((sf=3) and -2)
310       r(l)=r(l)-sf-sf-sf
320       r(l+1)=r(l+1)+sf
330       r(l-1)=r(l-1)+sf
340     }
350     if l>1 then y(l)=y(l)+py
360     if y(l)>y(l-1)-6 then y(l)=y(l-1)-6
370     if y(l)<y(l+1)+6 then y(l)=y(l+1)+6
380     if r(l)<10 then r(l)=10
390     if r(l)<>rr(l) or p(l)<>pp(l) or y(l)<>yy(l) then {
400       circle(256,yy(l),rr(l),0,180,360,100)
410       circle(256, y(l), r(l),c(p(l)),180,0,100)
420       rr(l)=r(l) :yy(l)=y(l) :pp(l)=p(l)
430     }
440   next
450 until exec(_bitsns,7) and 128
460 locate 0,2 :print "焼上げ中"
470 vpage(0)
480 for l=1 to 15
490   red=red+r(l)
500   grn=grn+y(l)
510 next
520 red=(red mod 23)+8
530 grn=(grn mod 23)+8
540 blu=(atoi(right$(time$,2)) mod 23)+8
550 for l=0 to 17
560   cl(l)=rgb(red-s(l),grn-s(l),blu-s(l))
570 next
580 circle(256,y(15),r(15),rgb(red-7,grn-7,blu-7),0,360,100)
590 paint(256,y(15),rgb(red-7,grn-7,blu-7))
600 for l=1 to 14
610   locate 10,2 :print using "##/14";l-1
620   le=y(l)-y(l+1)
630   fr=(r(l+1)-r(l))/(le*1#)
640   for ll=0 to le
650     yyy=y(l)-ll
660     rrr=r(l)+fr*ll
670     for lll=0 to 17
680       circle(256,yyy  ,rrr,cl(lll),sr(lll),er(lll),100)
690       circle(255,yyy  ,rrr,cl(lll),sr(lll),er(lll),100)
700       circle(256,yyy-1,rrr,cl(lll),sr(lll),er(lll),100)
710       circle(255,yyy-1,rrr,cl(lll),sr(lll),er(lll),100)
720     next
730   next
740 next
750 crt(1)
760 print "シフトキーとコントロールキーを押して下さい。"
770 repeat
780 until 3=exec(_bitsns,14)
790 vpage(1) :cls
```

リスト3 BDSELECT.S

```
 1: *        EXCONFIG用/起動ディレクトリセレクター
 2: *        BDSELECT.R                          (C)1994 ORE
 3:          .include        doscall.mac
 4:          .include        iocscall.mac
 5:          .text
 6:          .even
 7: *---------------------------------------------------------------
 8:                  *
 9:          tst.b   (a2)+
10:          beq     pcmopen          パラメータがない
11:                  *                パラメータのポインタを得る
12:          moveq.l #3,d7
13:          lea.l   ptrtable(pc),a6
14:          move.l  a2,(a6)+
15: nextptr          *                次のパラメータへ
16:          addq.w  #1,a2
17:          move.b  (a2),d0
18:          beq     keytst           ない
```

▶ああっ，ダンプ打ち込み中毒の禁断症状が……。　小川　貴也(15)千葉県

```
19:          cmpi.b  #' ',d0
20:          beq     clrspace
21:          cmpi.b  #9,d0
22:          bne     nextptr
23: clrspace         *                  空白消去
24:          clr.b   (a2)+
25: skipspace        *                  空白をスキップ
26:          move.b  (a2),d0
27:          cmpi.b  #' ',d0
28:          beq     clrspace
29:          cmpi.b  #9,d0
30:          beq     clrspace
31: ptrput           *                  ポインタを待避
32:          move.l  a2,(a6)+
33:          dbra    d7,nextptr
34: keytst           *                  XFキー押下テスト
35:          lea.l   ptrtable(pc),a6
36:          moveq.l #$0A,d1
37:          IOCS    _BITSNS
38:                  *                  XF1
39:          btst.l  #5,d0
40:          bne     chdir
41:                  *                  XF2
42:          addq.w  #4,a6
43:          btst.l  #6,d0
44:          bne     chdir
45:                  *                  XF3
46:          addq.w  #4,a6
47:          btst.l  #7,d0
48:          bne     chdir
49:                  *                  XF4
50:          moveq.l #$0B,d1
51:          IOCS    _BITSNS
52:          addq.w  #4,a6
53:          btst.l  #0,d0
54:          bne     chdir
55:                  *                  XF5
56:          addq.w  #4,a6
57:          btst.l  #1,d0
58:          beq     pcmopen            なにも押されていない
59: chdir            *                  ディレクトリ変更
60:          move.l  (a6),d0
61:          beq     pcmopen            キー押下に対するパス指定がない
62:          move.l  d0,-(sp)
63:          DOS     _CHDIR
64:          addq.w  #4,sp
65: pcmopen          *                  PCMファイルを開く
66:          clr.w   -(sp)
67:          pea.l   pcmfile(pc)
68:          DOS     _OPEN
69:          addq.w  #6,sp
70:          move.l  d0,d1              d1.w ファイルハンドル
71:          bmi     exit               ない場合はならさない
72:                  *                  PCMデータのサイズを得る
73:          move.w  #2,-(sp)
74:          pea.l   0
75:          move.w  d1,-(sp)
76:          DOS     _SEEK
77:          move.l  d0,d2              d2.l PCMデータのサイズ
78:          clr.w   6(sp)
79:          DOS     _SEEK
80:          addq.w  #8,sp
81:                  *                  PCMデータの読み込み
82:          move.l  d2,-(sp)
83:          movea.l 8(a0),a1
84:          addq.l  #1,d2
85:          andi.b  #-2,d2
86:          suba.l  d2,a1
87:          move.l  a1,-(sp)
88:          move.w  d1,-(sp)
89:          DOS     _READ
90:          addq.w  #6,sp
91:          move.l  (sp)+,d2
92:          tst.l   d0
93:          bmi     exit
94:                  *                  ファイルを閉じる
95:          move.w  d1,-(sp)
96:          DOS     _CLOSE
97:          addq.w  #2,sp
98: makearay         *                  チェーンテーブルの作成
99:          movea.l 8(a0),a2
100:         move.l  #$8000,d0
101:         divu.w  d0,d2
102:         swap.w  d2
103:         suba.w  d2,a2
104:         move.w  d2,-(a1)
105:         move.l  a2,-(a1)
106:         clr.w   d2
107:         swap.w  d2
108:         move.w  d2,d1
109:         addq.w  #1,d2
110:         subq.w  #1,d1
111:         bmi     pcmplay
112: nextaray
113:         move.w  d0,-(a1)
114:         adda.w  d0,a2
115:         move.l  a2,-(a1)
116:         dbra    d1,nextaray
117: pcmplay          *                  PCMデータの再生
118:         move.w  #4*256+3,d1
119:         IOCS    _ADPCMAOT
120: exit             *                  終了
121:         move.w  #-1,-(sp)
122:         DOS     _EXIT2
123: *-------------------------------------------------------------------
124: ptrtable         *                  パラメータのポインタテーブル
125:         .dc.l   0
126:         .dc.l   0
127:         .dc.l   0
128:         .dc.l   0
129:         .dc.l   0
130: pcmfile          *                  PCMファイル名
131:         .dc.b   'FANFARE.PCM',0
132:         .end
```

今月のもう一歩でした

今月から，投稿されたものの中から，おしくも採用されなかったプログラムをちょっとだけ紹介していきます。これを参考にして次回がんばってくださいね。また，面白いアイデアとか，作りかけのプログラムでうまくいかないことがある，などの悩みを抱えている人も，遠慮なく「(で)のショートプロ」まで投稿してみてください。できるかぎり皆さんのご要望にお応えしたいと思っています。

●B_MEN.C for X680x0

東京都　小平覚

このプログラムは2人対戦戦闘シミュレーションゲームで，各キャラに名前や属性を設定できるなど非常に豊富なパラメータとそれを生かしたゲーム展開がウリ。

実際，シミュレーション戦闘のモードに入ると足の速い盗賊がすすっと動いて，戦士に攻撃を加えたり，設定どおりにキャラが動いてくれる快感はありました。

ただ，問題となったのは基本的なプログラムのエラー処理です。メンバー設定中にキャンセルできなかったり，ユーザーデータをロードさせるのに，名前を間違えるとバスエラーを起こしたりとエラー処理が足りなすぎて，遊ぶのに非常に怖い思いをすることになってしまいました。短いリストでもエラーメッセージを出して止まるくらいはできたと思うのですが……。短いリストのわりにはかなり健闘していたプログラムだったので，致命的なエラー処理だけでもできていたら採用になっていたかもしれません。ちょっと残念。

●LRWAR.BAS for X680x0

岩手県　佐々木崇

あ，掲載作品もこの人だった。ま，いいか。このLRWARはXSPRITE.FNCを生かした対コンピュータ戦場シミュレーションゲーム。歩兵，戦車，戦闘機をテンキーで繰りだし，味方を残して敵を全滅させます。

このプログラムは兵士たちの動きが非常にユニークで，じっと見ていると蟻の軍隊を見ているような錯覚に陥ってしまいました。なんというか，戦いの無情感を感じさせるというか，非常に不思議な感じのする，奇妙に魅力的なゲームです。

ただ，実際にゲームを遊んでいても，なにが勝負の勝敗を分けるかよくわからない，という欠点があって，キーをひたすら叩いていれば勝ててしまったり，勝っても負けてもちょっと納得がいかないという困ったことになってしまいました。

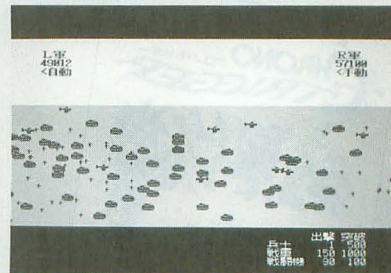
もう少し戦略性やかけひきが，きちんとわかるようにゲームが構成されていればよかったのではないかと思うのですが……また次回に期待しています。

B_MEN.C

LRWAR.BAS

▶6月号はX68000 PRO/PROIIユーザーには死刑宣告に近いものがありました。
八亀　桂一(20)神奈川県

Oh!X LIVE in '95

X68000・Z-MUSICver.2.0用
(SC-55+CM-32P対応)

©1995スクウェア「クロノ・トリガー」より

クロノ・トリガー

Ueda Hiroshi 上田 浩司

X68000・Z-MUSICver.2.0用
(CM-64対応)

「生命40億年はるかな旅」より

Planet of Life

Kishimoto Hideaki 岸本 英昭

X68000・Z-MUSICver.2.0用

©Nintendo「SUPER MARIO BROS.」より

SUPER MARIO BGM集

Shindo Noriyuki 進藤 慶到

今月は，ちょっとぜいたくに音源2台を使ったクロノ・トリガー，シンプルなピアノ曲のPlant of Life，そして懐かしのスーパーマリオです。進藤君がFM音源では難物とされるPSGのコピーを波形メモリを駆使して行っています。

クロノ・トリガー

「坂口博信，堀井雄二，鳥山明の3大クリエイターが手がける今世紀最高のRPG」というキャッチコピーで登場したスーパーファミコン用ソフト。ゲームの人気もさることながら「ファイナルファンタジー」シリーズのスクウェアが出した作品ということでBGMにも相当な期待が寄せられました。現在CD3枚組のオリジナルサウンドトラック(CD:PSCN5021-3/NTT出版)が発売されており，今月25日にはアレンジバージョンアルバムも発売されます。

さて，さっそくOh!X LIVEへも「クロノ・トリガー」第1号が投稿されてきました。曲目はそのものズバリ「クロノ・トリガー」，テレビCMにも流れていたメインテーマです。ゲームを知らなくても耳にしたことがある人も多いと思います。この曲はまさしくメインテーマ（主題）というにふさわしい曲で，そのメロディはゲーム中にさまざまなかたちで登場します。

このように，ひとつの決まったメロディを調を変えたり，リズムやコードを変えたりしてアレンジしながら使い回すことを，変奏，変奏曲といいます。こういった手法は映画音楽や歌劇にも見られるもので，その作品に統一した世界観を植えつけることができます。覚えておきましょう。

クロノ・トリガー

さて，このデータはなかなかの力作で楽器はCM-64とSC-55の2種類を使用します。ただし，CM-64はPCMパートしか使用していないのでCM-32Pで代用もできます。MIDIパッチベイなどを持っていない方は，

[X68000のMIDI OUT]→
[SC-55のMIDI IN]→
[SC-55のMIDI THRU]→
[CM-64/CM-32PのMIDI IN]

のように各機器をMIDIケーブルで接続してください。各楽器の出力は，ミキサーを持っていない方は，

[CM-64/CM-32Pの音声出力]→
[SC-55の音声入力]

のように接続してください。この状態で入力したリスト1を，

```
A>ZMUSIC
A>ZP filename
```

とすれば演奏開始されるはずです。

たびたびいっていることですがリスト中の行番号は単なる目安ですので入力してはいけません。

SC-55部分はSC-55mkII/SC-88でも正常な演奏を確認しています。

Planet of Life

次はピアノ曲です。チャンネル構成も2チャンネルと少なく，リストも短いので，ちょっと時間ができたときにでも打ち込めるサイズです。

曲目はNHKスペシャル「生命40億年はるかな旅」より「Planet of Life (Piano Solo)」。

ピアノ曲といってもショパンのような技巧的な曲ではなく，きわめてリリカルなメロディアスソングです。実に涼しげで夏にハマるというか……木陰で読書でもしながら聞いていたい，そんなイメージが湧きます。

一応CM-64用ですが，各楽器用の初期化メッセージなどを用意すればあっという間にGM音源用，GS音源用のデータに変身できます。

具体的には，

●GM音源の場合

・1トラック先頭の@iコマンド，Xコマンドを削除する。

・トラック1の先頭に，

@x$f0,$7e,$7f,$09,$01,$f7

を記述する（GMシステムON）。

・音色を指定している「@5」の部分をすべて「@1」に変更する。

●GS音源の場合

・1トラック先頭の@iコマンド，Xコマン

ドを削除する。
・リスト先頭の「(i)」と「.comment」の間に「.sc55_init」を挿入する。
・音色を指定している「@5」の部分をすべて「@1」に変更する。
とこんな感じです。

このように複雑な構成でないものならば他機種用の曲はもちろん内蔵音源用の曲もかなり容易にMIDI楽器用に作り変えることができます。いろいろ挑戦してみましょう。　(Z.N)

近頃ファミコンにハマってます

今回はファミコン版「SUPER MARIO BROS.」をOPMで再現してみました。

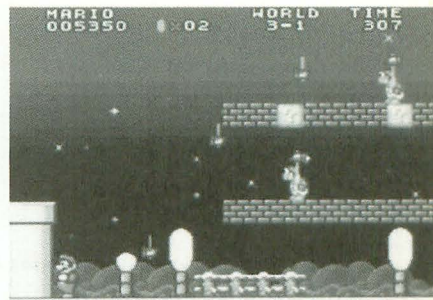
マリオというキャラクターはあちこちに登場していますから，知らない人はほとんどいないでしょう。この兄弟が世界にはばたくきっかけとなったのが「SUPER MARIO BROS.」への出演でした。私はこのゲームで育った世代ですが，皆さんはどうでしょうか？　興味が湧いたならぜひ入力して，一世を風靡したファミコンサウンドを思い出してみてください。あぁ，やっぱこういう音がいちばん気持ちいいよなぁ……。

以前，MSX版グラ2エンディングをやりましたが，今回はそれよりもさらにチープな音源からのコピーです。さすがに，手を抜くとすぐバレますし，なにより自分自身が納得できん……ということで，できうる限り原曲に近づけたつもりです。

最近はこういうのばかり作って楽しんでまして，ここにきてファミコンに落ち着いてます。比較的手軽に作れるのがよし。「このことにどういう意味が？」という意見もあるでしょうが，なんというか，これは私にとって模型を作る感覚に限りなく近いものです。私はプラモ歴もあるんですが，完成品を眺めることよりも，作るという行為そのものがいちばんの動機であり目的でした。それを音でやったということですね。今回のデータなら，ノートごとのディチューンや出力レベルなど，まだまだ似せる要素は多く，模型でいえばプラモに彩色した程度になりますか（接着ライン処理くらいはいくか？）。

さて，リストは全部で3つあります。それぞれにつけたタイトルは「地上」「水中」「終了」となっていますが，「終了」のみドラマ形式の進行にしてみました。

ファミコンの生音を再現したことから，RF音声より遥かにシャープなサウンドになりました。ちょっとノイジーですね。

PSGのコピーに不可欠なソフトエンベロープは，ユーザー波形で再現しています。いまのところ波形データはすべて手作業で書いているんですが，やはり専用エディタ

SUPER MARIO BROS.（スーパーファミコン版）

がほしくなる……。演奏ビュアも，波形の情報が詳しく反映されるものが望まれます。これからは波形の時代ですから，サポートツールの開発を予定されている方は，頭の隅にでも入れておいてくれれば幸いです。

実はこれ，私の趣味だけを追求したデータってことで，誌上で発表するつもりはまったくなかったものです。なんだかんだで掲載に至ったわけですが，ZMUSIC.Xのバージョンによる発音の違いは解消できませんでした（そんなに大袈裟な差ではないが）。できればver.2.04以上で再生してください（なお最新版は来月号の付録ディスクに収録予定）。

ファミコンのコピーなんて今回が最後でしょうが，こういうネタのストックはたくさんありますので，もし気に入っていただけたらリクエストなど送ってくださいね。ではまた。　(進藤慶到)

リスト1　クロノ・トリガー

```
==============  CRO.ZMS  ==================
 1: .comment -CHRONO TRIGGER-TITLE-1995 SQUARE SOFT-
 2: / composer SQUARE      3: / programmer h.U 1995 03/22
 4: / for ZMUSIC.X Ver2.0
 5: / MIDI MODULE : CM32P+SC-55
 6: /
 7: /************************************************************
 8:
 9: /TRACK SET UP************************************************
10:
11: (i)
12: (b0)
13:
14: (m1,5000)(aMIDI9,1)    /CM32P Bass
15: (m2,5000)(aMIDI12,2)   /CM32P STRINGS 1
16: (m3,5000)(aMIDI13,3)   /CM32P STRINGS 2
17: (m4,5000)(aMIDI14,4)   /CM32P STRINGS 2
18: (m5,5000)(aMIDI1,5)    /SC-55 Melody
19: (m6,5000)(aMIDI2,6)    /SC-55 Melody
20: (m7,5000)(aMIDI3,7)    /SC-55 Melody D
21: (m8,5000)(aMIDI15,8)   /CM32P
22: (m9,5000)(aMIDI4,9)    /SC-55 Harp
23: (m10,5000)(aMIDI5,10)  /SC-55
24: (m11,5000)(aMIDI6,11)  /SC-55 Timpani
25: (m12,5000)(aMIDI7,12)  /SC-55 Timpani
26: (m13,5000)(aMIDI8,13)  /SC-55 etc
27: (m14,5000)(aMIDI10,14) /SC-55 drums
28: (m15,5000)(aMIDI10,15) /SC-55 drums
29: (m16,5000)(aMIDI11,16) /SC-55 drums
30: (m17,5000)(aMIDI11,17) /SC-55 drums
31: (m18,5000)(aMIDI16,18) /CM32P Melody
32:
33:
34: /MIDI SYSTEM SET UP*****************************************
35: /SC-55 System set up
36:
37: .roland_exclusive $10,$42={$40,$00,$7f,$00}
38: .roland_exclusive $10,$42={$40,$01,$30,$04}
39: .roland_exclusive $10,$42={$40,$01,$38,$02}
40:
41: .SC55_part_SETUP  1,$10 ={01}
42: .SC55_part_SETUP  2,$10 ={02}
43: .SC55_part_SETUP  3,$10 ={03}
44: .SC55_part_SETUP  4,$10 ={04}
45: .SC55_part_SETUP  5,$10 ={05}
46: .SC55_part_SETUP  6,$10 ={06}
47: .SC55_part_SETUP  7,$10 ={07}
48: .SC55_part_SETUP  8,$10 ={08}
49: .SC55_part_SETUP  9,$10 ={17}
50: .SC55_part_SETUP 10,$10 ={10}
51: .SC55_part_SETUP 11,$10 ={11}
52: .SC55_part_SETUP 12,$10 ={17}
53: .SC55_part_SETUP 13,$10 ={17}
54: .SC55_part_SETUP 14,$10 ={17}
55: .SC55_part_SETUP 15,$10 ={17}
56: .SC55_part_SETUP 16,$10 ={17}
57:
58: .SC55_v_reserve $10={2,2,2,2,2,2,2,2,0,4,4,0,0,0,0,0}
59:
60:
61: /CM-32P System set up
62:
63: .roland_exclusive 16,22={
64:                               $52,0,1
65:                               2,4,5
66:                               2,3,3,9,6,3
67:                               08,11,12,13,14,15}
68: /MT-32
69: .roland_exclusive 16,22={
70:                               $10,0,1
71:                               1,1,1
72:                               1,1,1,1,1,1,1,1,1
73:                               17,17,17,17,17,17,17,17,17}
74:
75: /MML DATA***************************************************
76:
77: (t1)  t138 r2
78: (t2)  t138 r2
79: (t3)  t138 r2
80: (t4)  t138 r2
81: (t5)  t138 r2
82: (t6)  t138 r2
```

```
 83: (t7)  t138 r2
 84: (t8)  t138 r2
 85: (t9)  t138 r2
 86: (t10) t138 r2
 87: (t11) t138 r2
 88: (t12) t138 r2
 89: (t13) t138 r2
 90: (t14) t138 r2
 91: (t15) t138 r2
 92: (t16) t138 r2
 93: (t17) t138 r2
 94: (t18) t138 r2
 95:
 96: /01w Bass
 97: (t1) @28 q8 o1 l8 v12 @u110 @k0 @p58 @k2 r2
 98: (t1) r2
 99: (t1) q8araarq7aaa q8araarq7aaa
100: (t1) q8f+rf+f+rq7f+f+f+ q8f+rf+f+rq7f+f+f+
101: (t1) q8frffrq7fff q8frffrq7fff
102: (t1) q8ereerq7eee q8ereerq7eee
103: (t1) |:2
104: (t1) |:2q8araarq7aaa q8araarq7aaa
105: (t1) q8f+rf+f+rq7f+f+f+ q8f+rf+f+rq7f+f+f+:|
106: (t1) q8frffrq7fff q8frffrq7fff
107: (t1) q8ereerq7eee q8ereerq7eee
108: (t1) q8crccrq7ccc q8frffrq7fff
109: (t1) q8erq7eeeeq8e4 ere e<e>e ee
110: (t1) :| @h144 @m90  @u120 q8
111: (t1) g1a1b*288a2 g1f+1b*288a2
112: (t1) g1a1b*278&@b0,-1365,0b*10@b0a2 g1f+1 @m0
113: (t1) @u110 |:2 c.g.<c>c2 c.c.cc4<c>c
114: (t1)            d.a.<d>d2 d.a.<ded>ad :|
115: (t1) @u120 q6 l16 eere eree reer eere  er2...
116:
117: /CM32P STR 1
118: (t2) @35 q8 l8 v08 @u110 @k0 @k-1 r2
119: (t2) r2 o3
120: (t2) @p54 @u100 b1<e2a2 b2<e2 a2b2 <@u110c1 d2e2 f+8.g8.f+1
6d16>@u100b2& b1
121: (t2) r*6048
122: (t2) @p50 @u100>a8.<e8f+16a8 @u90
123: (t2) b2..a16g16 f+2d2 c+8.d8.a8&a2&a1 b2.ab <c+2d2> b1&b2>
@u100a8.<e8f+16a8 @u90
124: (t2) b2..a16g16 f+2d2 c+8.d8.a8&a2&a1 b2.ab <c+2d2>
125: (t2) l16     q8
126: (t2) v04 |:15b&~3:|b   v06 |:15b&~1:|b <
127: (t2) v06 |:15d&~1:|d   v06 |:15d&~1:|d
128: (t2) v06 |:15e&~1:|e   v06 |:15e&~1:|e
129: (t2) v06 |:15f+&~1:|f+ v06 |:15f+&~1:|f+
130: (t2) v08 @u105 @p64 q7 l16 o5
131: (t2) l16 bbrb brbb rbbr bbrb br
132:
133: /01w str 2
134: (t3) @35 q8 l8 v06 @u100 @k0 @p74 @k1 r2 o4
135: (t3) r2 @u90
136: (t3) 'b<e*384' 'a<c+*384' '<ce*384' 'b<d*384'
137: (t3) o4 q4 @u115 @p92
138: (t3) |:2
139: (t3) |:2'a<ce'r4'a<ce'r4.'b<df+':|  |:2'a<c+e'r4'a<c+e'r4.'
ef+b':|
140: (t3) |:2'a<ce'r4'a<ce'r4.'b<df+':|  |:2'a<c+e'r4'a<c+e'r4.'
ef+b':|
141: (t3) |:2'g<ce'r4'g<ce'r4.'dgb'  :|  |:2'b<df+'r4'b<df+'r4.'
<ceg':|
142: (t3)    'g<ce'r4'g<ce'r4.'b<eg'        '<cfa'r4'<cfa'r4.'b<
eg'
143: (t3)    'a<df+'r*120'b<df+'r*120   |q6 'a<df+'r16 'a<df+'r1
6 q7'a<dg16''a<dg16'
144: (t3) :| r2
145: (t3) o5 q8 @u90  @p74
146: (t3) d1f+1d*384  f+1e1'>b<f+*384'
147: (t3) d1f+1d*384  f+1e1
148: (t3) l16 q6 q6 @u110 @p94 v08
149: (t3) eree reer errr rrer    eree reer errr rrer
150: (t3) f+rf+f+ rf+f+r f+rrr rrf+r
151: (t3) f+rf+f+ rf+f+r f+rf+r @u120g@u110f+@u120g@u110f+
152: (t3) eree reer errr rrer    eree reer errr rrer
153: (t3) f+rf+f+ rf+f+r f+rrr rrf+r
154: (t3) f+rf+f+ rf+f+r f+rf+r @u120drdr
155: (t3) v08 @u105 @p74 q7 l16 o5
156: (t3) ddrd drdd rddr ddrd dr
157:
158: /01w str 2
159: (t4) @35 q8 l8 v06 @u100 @k0 @p84 @k0 r2 o5
160: (t4) r2
161: (t4) g*384 e*384 a*384 f+1g2.. @59 r8
162: (t4) o4 q4 @u112 @p44
163: (t4) |:2
164: (t4) |:2'a<ce'r4'a<ce'r4.'b<df+':|  |:2'a<c+e'r4'a<c+e'r4.'
ef+b':|
165: (t4) |:2'a<ce'r4'a<ce'r4.'b<df+':|  |:2'a<c+e'r4'a<c+e'r4.'
ef+b':|
166: (t4) |:2'g<ce'r4'g<ce'r4.'dgb'  :|  |:2'b<df+'r4'b<df+'r4.'
<ceg':|
167: (t4)    'g<ce'r4'g<ce'r4.'b<eg'        '<cfa'r4'<cfa'r4.'b<
eg'
168: (t4)    'a<df+'r*120'b<df+'r*120   |q6 'a<df+'r16 'a<df+'r1
6 'a<dg16''a<dg16'
169: (t4) :| r4 @35 r4
170: (t4) o4 q8 @u90  @p54
171: (t4) b1<c+1>a1&a1   a1f+1<e1d+1>
172: (t4) b1<c+1>a1&a1   a1f+1
173: (t4) l16    q4
174: (t4) l16 q6 q6 @u110 @p34 v08
175: (t4) brbb rbbr brrr rrbr    brbb rbbr brrr rrbr
176: (t4) araa raar arrr rrar    araa raar arar arar
177: (t4) brbb rbbr brrr rrbr    brbb rbbr brrr rrbr
178: (t4) araa raar arrr rrar    araa raar arar arar
179: (t4) v08 @u105 @p54 q7 l16 o5
180: (t4) o5 f+f+rf+ f+rf+f+ rf+f+r f+f+rf+ f+r
181:
182: /SC-55 Melody
183: (t5) @i$41,$10,$42 @50 q8 l16 v15 @u110 @k1 @p63 @e60,00 r2
184: (t5) r2 o3
185: (t5) @p64 @u80 b1<e2a2 b2<e2 a2b2 <@u90c1 d2e2 f+8.g8.f+16d
16>@u80b2& b2. @h48 @m70
186: (t5) @57 @k-1 @p44 q8 r8 o4 @q1 @u80 b8<
187: (t5) |:2
188: (t5) e8.f+8.g8 >b8.<f+8.d8& d2r8 >b<df+a8.  b8.b8.ag+e2& e2
.r8>b8<
189: (t5) e8.f+8.g8 >b8.<f+8.d8& d2r8 >b<df+a8.  b8.b8.ag+e2& e2
.r8 ef+
190: (t5) g2a2 b4.<q7c4q8c4  @b0,1365,0c&c8.@b0 d4q8 c+c>b2& b2.
r8d8
191: (t5) e2g2 a2g2 | f+eq4f+8r2q8f+eq4f+8r2..q8>b8<
192: (t5) :|
193: (t5)              f+eq4f+8r2q8f+eq4f+8r1
194: (t5) r*2688
195: (t5) @u90 q8
196: (t5) <@b0,1365,0d&d2.r@b0de>  b8.a8.b8e4.ef+ g8.a8.gf+e8.d8
.>b8& b1
197: (t5) r8ab<cd8e8.f+8g8a8  b8.e8.<d4c4>b<c>   b8.a8.g8a2&   a
2g4f+4
198: (t5) q7 f+8.g8.f+8. d8.>b8.<e8.
199:
200: /SC-55 Melody
201: (t6) @i$41,$10,$42 @49 q8 l8 v15 @k-2 @p54 @e60,00 r2
202: (t6) r2 o6 @u60 @p104
203: (t6) g*384 e*384 a*384 f+1g2. @h48 @m70 l16
204: (t6) @57 @k01 @p84 q8 r8 o4 @q1 @u80 b8<
205: (t6) |:2
206: (t6) e8.f+8.g8 >b8.<f+8.d8& d2r8 >b<df+a8.  b8.b8.ag+e2& e2
.r8>b8<
207: (t6) e8.f+8.g8 >b8.<f+8.d8& d2r8 >b<df+a8.  b8.b8.ag+e2& e2
.r8 ef+
208: (t6) g2a2 b4.<q7c4q8c4  @b0,1365,0c&c8.@b0 d4q8 c+c>b2& b2.
r8d8
209: (t6) e2g2 a2g2 | f+eq4f+8r2q8f+eq4f+8r2..q8>b8<
210: (t6) :|
211: (t6)              f+eq4f+8r2q8f+eq4f+8r1
212: (t6) r*2688
213: (t6) @u90 q8
214: (t6) <@b0,1365,0d&d2.r@b0de>  b8.a8.b8e4.ef+ g8.a8.gf+e8.d8
.>b8& b1
215: (t6) r8ab<cd8e8.f+8g8a8  b8.e8.<d4c4>b<c>   b8.a8.g8a2&   a
2g4f+4
216: (t6) q7 f+8.g8.f+8. d8.>b8.<e8.
217:
218: /SC-55 Melody
219: (t7) @i$41,$10,$42 @45 q8 l16 v15 @u110 @k1 @p63 @e60,00 r2
220: (t7) r2 o4
221: (t7) @p24 @u40 b1<e2a2 b2<e2 a2b2 <@u90c1 d2e2 f+8.g8.f+16d
16>@u80b2& b2.  @h48 @m70
222: (t7) @57 @k03 @p64 q8 r8  r8 o4 @q1 @u65 b8<
223: (t7) |:2
224: (t7) e8.f+8.g8 >b8.<f+8.d8& d2r8 >b<df+a8.  b8.b8.ag+e2& e2
.r8>b8<
225: (t7) e8.f+8.g8 >b8.<f+8.d8& d2r8 >b<df+a8.  b8.b8.ag+e2& e2
.r8 ef+
226: (t7) g2a2 b4.<q7c4q8c4  @b0,1365,0c&c8.@b0 d4q8 c+c>b2& b2.
r8d8
227: (t7) e2g2 a2g2 | f+eq4f+8r2q8f+eq4f+8r2..q8>b8<
228: (t7) :|
229: (t7)              f+eq4f+8r2q8f+eq4f+8r1
230: (t7) r*2688
231: (t7) @u85 q8
232: (t7) <@b0,1365,0d&d2.r@b0de>  b8.a8.b8e4.ef+ g8.a8.gf+e8.d8
.>b8& b1
233: (t7) r8ab<cd8e8.f+8g8a8  b8.e8.<d4c4>b<c>   b8.a8.g8a2&   a
2g4f+4
234: (t7) q7 f+8.g8.f+8. d8.>b8.e16
235:
236: /CM32P
237: (t8) @62 q8 l16 v09 @u110 @k0 @k-1 r2
238: (t8) r2 o3
239: (t8) @p84 @u80 b1<e2a2 b2<e2 a2b2 <c1 d2e2 f+8.g8.f+16d16>b
2& b1
240: (t8) r*5952
241: (t8) r2 @11 o4 @u120 @k00 l8 v10 r2
242: (t8) |:2
243: (t8) @d1@u120b<@u105f+ef+@d0 @d1@u120b@u105f+ef+>@d0
244: (t8) @d1@u120a<@u105f+ef+@d0 @d1@u120a@u105f+ef+>@d0
245: (t8) @d1@u120<c+@u105dc+d@d0 @d1@u120f+@u105dc+d>@d0
246: (t8) @d1@u120<c+@u105dc+d@d0 @d1@u120f+@u105dc+d>@d0
247: (t8) @d1@u120b<@u105f+ef+@d0 @d1@u120b@u105f+ef+>@d0
248: (t8) @d1@u120a<@u105ec+e@d0  @d1@u120a@u105ec+|e> @d0
249: (t8) @d1@u120b<@u105f+ef+@d0 @d1@u120b@u105f+ef+>@d0
250: (t8) @d1@u120b<@u105f+d+f+>@d0 @d1@u120b<@u105f+d+@u100f+>@
d0
251: (t8) :| <e16 v09
252: (t8) @19 @u90 o2 @p68 r16
253: (t8)        |:2 c.g.<c>c2 c.c.cc4<c>c
254: (t8)            d.a.<d>d2 d.a.<ded>ad :|
255: (t8) @u105 q6 l16<eere eree reer eere er2...
256:
257: /SC-55
258: (t9) @i$41,$10,$42 @47 q8 l24 v10 @u110 @k0 @e90,30 r2
259: (t9) r2 o4 q3 @u70
260: (t9) r*1440 @p24f+b<e @p44>b<ef+ @p74ef+b @p94f+b<e
261: (t9) r*2880 o4 @u70
262: (t9) @p24ef+a @p34b<de> @p44b<de @p54f+ab @p74f+ab @p84<def
+ @p94def+ @p104ab<d
263: (t9) r*1536
264: (t9) o2 @p84 @h144 @m90 @u60  q8 @36 r*1536
265: (t9) g1a1b*288a2 g1f+1b*288a2
266: (t9) g1a1b*278&@b0,-1365,0b*10@b0a2 g1f+1
267: (t9) @02 @u115 o6 q6 l8
268: (t9) |:8@p34b@p94a@p44b@p84e @p34b@p94a@p44b@p84e:|
269: (t9) r1r1
270:
```


▶C言語よりもアセンブラのほうが理解しやすいと思うのは私だけでしょうか？
角谷　匠(23)富山県

```
271: /SC-55 STR 1
272: (t10) @i$41,$10,$42 @92 q8 l8 v08 @u110 @k0 @e120,00 r2
273: (t10) r2 o3
274: (t10) @p84 @u80 b1<e2a2 b2<e2 a2b2 <@u90c1 d2e2 f+8.g8.f+16
d16>b2& b1
275: (t10) r*6048
276: (t10) @p78 @u115>a8.<e8f+16a8
277: (t10) b2..a16g16 f+2d2 c+8.d8.a8&a2&a1 b2.ab <c+2d2> b1&b2>
 a8.<e8f+16a8
278: (t10) b2..a16g16 f+2d2 c+8.d8.a8&a2&a1 b2.ab <c+2d2>
279: (t10) l16     q8 @e40,00
280: (t10) v04 |:15b&¯3:|b   v06 |:15b&¯1:|b <
281: (t10) v06 |:15d&¯1:|d   v06 |:15d&¯1:|d
282: (t10) v06 |:15e&¯1:|e   v06 |:15e&¯1:|e
283: (t10) v06 |:15f+&¯1:|f+ v06 |:15f+&¯1:|f+
284: (t10) v09 @u110 @p64 q7 l16 o5
285: (t10) l16 bbrb brbb rbbr bbrb br
286:
287: /SC-55 Tim
288: (t11) @i$41,$10,$42 @48 q8 l8 v10 @u120 @k0 @e30,00 r2
289: (t11) o2 @p34 q4 r4.v12<e8>v09
290: (t11) arrarrra arrarrra <f+rrf+rrrf+ f+rrf+rrrf+
291: (t11) frrfrrrf frrfrrrf errerrre errerrre>
292: (t11) |:2
293: (t11) arrarrra arrarrra <f+rrf+rrrf+ f+rrf+rrrf+>
294: (t11) arrarrra arrarrra <f+rrf+rrrf+ f+rrf+rrrf+
295: (t11) frrfrrrf frrfrrrf errerrre    errerrre
296: (t11) crrcrrrc frrfrrrf >r4a16a16a8r2  a16a16a8r4<e8.>a8<e1
6e16e16>
297: (t11) :|
298: (t11) r*2688 l16 <
299: (t11) c2.g4 c8.c8.ccc4.c8  d2.a4 d4.dd drdr adad
300: (t11) c2.g4 c8.c8.ccc4.c8  d2.a4 d4.dd drdr adad
301: (t11) eere eree reer eere er
302:
303: /SC-55 tim
304: (t12) @i$41,$10,$42 @48 q8 l8 v10 @u120 @k1 @e30,00 r2
305: (t12) o2 @p94 q4 r4.v12<e8>v09
306: (t12) arrarrra arrarrra <f+rrf+rrrf+ f+rrf+rrrf+
307: (t12) frrfrrrf frrfrrrf errerrre errerrre>
308: (t12) |:2
309: (t12) arrarrra arrarrra <f+rrf+rrrf+ f+rrf+rrrf+>
310: (t12) arrarrra arrarrra <f+rrf+rrrf+ f+rrf+rrrf+
311: (t12) frrfrrrf frrfrrrf errerrre    errerrre
312: (t12) crrcrrrc frrfrrrf >r4a16a16a8r2  a16a16a8r4<e8.>a8<e1
6e16e16>
313: (t12) :|
314: (t12) r*2688 l16 <
315: (t12) c2.g4 c8.c8.ccc4.c8  d2.a4 d4.dd drdr adad
316: (t12) c2.g4 c8.c8.ccc4.c8  d2.a4 d4.dd drdr adad
317: (t12) eere eree reer eere er
318:
319: /SC-55
320: (t13) @i$41,$10,$42 @49 q8 l8 v15 @k2 @p54 @e60,00 r2
321: (t13) r2 o5 @u60 @p24
322: (t13) 'b<e*384' 'a<c+*384' '<ce*384' 'b<d*384'
323: (t13) o5 @u70 @p64 r*6144
324: (t13) 'b<d1''<c+f+1''a<d*384'  'a<f+1''f+<e1''>b<ef+1''>b<d
+f+1'
325: (t13) 'b<d1''<c+f+1''a<d*384'  'a<f+1''f+<e1'
326: (t13) l16 q8 @62 o4 @p64 @u110
327: (t13) v07 |:15b&¯3:|b   v09 |:15b&¯1:|b <
328: (t13) v09 |:15d&¯1:|d   v09 |:15d&¯1:|d
329: (t13) v09 |:15e&¯1:|e   v09 |:15e&¯1:|e
330: (t13) v09 |:15f+&¯1:|f+ v09 |:15f+&¯1:|f+
331: (t13) v12 @u110 @p64 q7 l16 o4
332: (t13) l16 bbrb brbb rbbr bbrb br
333:
334: /SC-55 Dr
335: (t14) @i$41,$10,$42 @49 q8 l16 v11 @k2 @p64 @e70,00 r2 o2
336: (t14) l24 r4 @u60d@u70d@u80d ddd
337: (t14) l16
338: (t14) |:6
339: (t14) @u90drdr @u80dd@u100dr         dr@u80d@u100d r@u80d@u1
00dr
340: (t14) @u90drdr @u50d24d24d24@u100dr dr@u80d@u100d r@u80d@u1
00d@u80d |
341: (t14) @u80dd@u100dr @u80dd@u100dr   dr@u80d@u100d r@u80d@u1
00d@u80d
342: (t14) @u90drdr @u50d24d24d24@u100dr dr@u80d@u100d r@u80d@u1
00d@u80d
343: (t14) :|
344: (t14) @u90dddr r2 ddd8 r2 @u60d24d24@u70d24 d24@u80d24d24 @
u90dd@u100d8
345: (t14) |:4
346: (t14) @u90drdr @u80dd@u100dr         dr@u80d@u100d r@u80d@u1
00dr
347: (t14) @u90drdr @u50d24d24d24@u100dr dr@u80d@u100d r@u80d@u1
00d@u80d |
348: (t14) @u80dd@u100dr @u80dd@u100dr   dr@u80d@u100d r@u80d@u1
00d@u80d
349: (t14) @u90drdr @u50d24d24d24@u100dr dr@u80d@u100d r@u80d@u1
00d@u80d
350: (t14) :|
351: (t14) @u90dddr r2 ddd8 r2 @u60d24d24@u70d24 d24@u80d24d24 @
u90dd@u100d8
352: (t14) r*2304
353: (t14) @u40d4@u60d4@u80d4@u90d4  d4d4drdr @u100drdd
354: (t14) |:4
355: (t14) @u80drdd rddd drdr @u20d24@u30d24@u40d24@u60d@u70d |
356: (t14) @u80drdd rddd drdr @u90dr@u80dd
357: (t14) :|
358: (t14) @u80drdd rddd drdr @u40d24@u50d24@u60d24 @u80d@u90d
359: (t14) ddrd drdd rddr ddrd dr
360:
361: /SC-55 Dr
362: (t15) @i$41,$10,$42 @49 q8 l8 v11 @k2 @p64 @e70,00 r2 o1
363: (t15) r2 @u85
364: (t15) |:8brrb rrrb:|
365: (t15) |:14brrb rrrb:| bbr2bb b4r2.
366: (t15) |:14brrb rrrb:| bbr2bb b4r2.
367: (t15) o4 l16
368: (t15) |:56 @u120a@u50a@u110a@u120a :|
369: (t15) o1 @u70
370: (t15) |:8b8.b8.b8b4b8b8:|b8.b8.b8.b8.b8.bbr r2..
371:
372: /SC-55 Dr
373: (t16) @i$41,$10,$42 x$40,$1a,$15,$02 @17 q8 l8 v07 @k2 @p64
 @e20,10 r2 o3
374: (t16) r2  @u120
375: (t16) 'c+a4'r*1488
376: (t16) |:2 @u110
377: (t16) 'c+a4'r*2640
378: (t16) c+4r2a4 r1
379: (t16) :|  @u115
380: (t16) 'c+a4'r*1488
381: (t16) 'c+a4'r*720 @u100>c4c4c4c4 c4c4c4c4< @u110
382: (t16) |:4'c+a4'r2.r1:|r1r1
383:
384: /SC-55 Dr
385: (t17) @i$41,$10,$42 x$40,$1a,$15,$02 @17 q8 l8 v07 @k2 @p64
 @e20,10 r2 o2
386: (t17) l24 r4 @u60d@u70d@u80d ddd
387: (t17) l16
388: (t17) |:6
389: (t17) @u90drdr @u80dd@u100dr         dr@u80d@u100d r@u80d@u1
00dr
390: (t17) @u90drdr @u50d24d24d24@u100dr dr@u80d@u100d r@u80d@u1
00d@u80d |
391: (t17) @u80dd@u100dr @u80dd@u100dr   dr@u80d@u100d r@u80d@u1
00d@u80d
392: (t17) @u90drdr @u50d24d24d24@u100dr dr@u80d@u100d r@u80d@u1
00d@u80d
393: (t17) :|
394: (t17) @u90dddr r2 ddd8 r2 @u60d24d24@u70d24 d24@u80d24d24 @
u90dd@u100d8
395: (t17) |:4
396: (t17) @u90drdr @u80dd@u100dr         dr@u80d@u100d r@u80d@u1
00dr
397: (t17) @u90drdr @u50d24d24d24@u100dr dr@u80d@u100d r@u80d@u1
00d@u80d |
398: (t17) @u80dd@u100dr @u80dd@u100dr   dr@u80d@u100d r@u80d@u1
00d@u80d
399: (t17) @u90drdr @u50d24d24d24@u100dr dr@u80d@u100d r@u80d@u1
00d@u80d
400: (t17) :|
401: (t17) @u90dddr r2 ddd8 r2 @u60d24d24@u70d24 d24@u80d24d24 @
u90dd@u100d8
402: (t17) r*2304
403: (t17) @u40d4@u60d4@u80d4@u90d4  d4d4drdr @u100drdd
404: (t17) |:4
405: (t17) @u80drdd rddd drdr @u20d24@u30d24@u40d24@u60d@u70d |
406: (t17) @u80drdd rddd drdr @u90dr@u80dd
407: (t17) :|
408: (t17) @u80drdd rddd drdr @u40d24@u50d24@u60d24 @u80d@u90d
409: (t17) ddrd drdd rddr ddrd dr
410:
411: /CM-32P
412: (t18) @31 q8 l16 v10 @u110 @k1 r2
413: (t18) r2 o3
414: (t18) @p100 @u70 b1<e2a2 b2<e2 a2b2 <@u90c1 d2e2 f+8.g8.f+1
6d16>@u80b2& b2. @h48 @m70
415: (t18) @47 @k02 @p64 q8 r8 o3 @q1 @u65 b8<
416: (t18) |:2
417: (t18) e8.f+8.g8 >b8.<f+8.d8& d2r8 >b<df+a8.  b8.b8.ag+e2& e
2.r8>b8<
418: (t18) e8.f+8.g8 >b8.<f+8.d8& d2r8 >b<df+a8.  b8.b8.ag+e2& e
2.r8 ef+
419: (t18) g2a2 b4.<q7c4q8c4  @b0,1365,0c&c8.@b0 d4q8 c+c>b2& b2
.r8d8
420: (t18) e2g2 a2g2 | f+eq4f+8r2q8f+eq4f+8r2..q8>b8<
421: (t18) :|
422: (t18)                 f+eq4f+8r2q8f+eq4f+8r1
423: (t18) r*2688
424: (t18) @u75 q8
425: (t18) <@b0,1365,0d&d2.r@b0de>  b8.a8.b8e4.ef+ g8.a8.gf+e8.d
8.>b8& b1
426: (t18) r8ab<cd8e8.f+8g8a8  b8.e8.<d4c4>b<c>   b8.a8.g8a2&
a2g4f+4
427: (t18) q7 f+8.g8.f+8. d8.>b8.<e8.
428:
429:
430: (p)
431: /END
```

リスト2 クロノ・トリガーカウンタ表示

```
 1:00003120 00000000   2:00003078 00000000   3:00003078 00000000   4:00003078 00000000
 5:00003078 00000000   6:00003078 00000000   7:00003078 00000000   8:00003120 00000000
 9:00003120 00000000  10:00003078 00000000  11:00003078 00000000  12:00003078 00000000
13:00003078 00000000  14:00003078 00000000  15:00003120 00000000  16:00003120 00000000
17:00003078 00000000  18:00003078 00000000
```

▶瀧氏が学生らしいということに気がついた。大学ってものすごいところなんですね。
林　大助(19)神奈川県

リスト3 Planet of Life

```
==================  PLANET_P.ZMS  ==================
 1: /*□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□
 2: /*□      NHKスペシャル「生命40億年はるかな旅」より        □
 3: /*□            『Planet of Life (Piano Solo)』              □
 4: /*□                      作曲:大島ミチル                  □
 5: /*□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□
 6: (i)
 7: .comment Planet of Life  生命40億年はるかな旅より for CM-64
 8:
 9: (m1,4000)(aMIDI11,1)
10: (m2,4000)(aMIDI12,2)
11:
12: (t1) @i65,16,22x127,0,0,0x16,0,22,100x82,0,16,100t63
13: (t1) x82,0,0,64,0,5,5r32@5@b0@v105y11,127r3214p3
14: (t1) |:t63o5r@u74@q1c@q2a+.@q0a8 @u66ccr2
15: (t1) r@u84@q1c<@u86d.c8 t62@u84c>@u72f2r
16: (t1) rft64<@u75@q5f.@q0@u62c+8 @u70c218r>@u74e@u57f@u67g |
17: (t1) a2r8@u63d@u55e@u63f18 t62@u51g2t58@u31'cea'96,0:|
18: (t1) 14t64@u78a2r8@u73g+8@u84a8@u87b8 <@u99e2.r>
19: (t1) r@u90a@q6g.@q0@u65f8 @u79e>@u67a2r<
20: (t1) r@u87g@q6f.@q0@u77d+8
21: (t1) @u72d>@q4g.r8@q0@u98g*18@u85g+*15a+*15
22: (t1) @u89<c2r@u101>g+*18@u91a+*15<c*15 d2r>@u77g
23: (t1) t62@u59'a+<dg'@u40'a+<d'@u34'a+<d't61'a+<d'
24: (t1) t59@u46'a<cea',3rt56@u31t52'ga+<ceg',4t50r
25: /* 以下4行は | の位置以外上と同じ
26: (t1) |:t63o5r@u74@q1c@q2a+.@q0a8  @u66ccr2 |
27: (t1) r@u84@q1c<@u86d.c8 t62@u84c>@u72f2r
28: (t1) rft64<@u75@q5f.@q0@u62c+8 @u70c218r>@u74e@u57f@u67g
29: (t1) a2r8@u63d@u55e@u63f18 t62@u51g2t58@u31'cea'96,0:|
30: (t1) 14@u57c@u79@q1c<@u80d.c8 f>@u78fr8t62@u70e8@u56f8@u71g8
31: (t1) t63@u61a.r8<@u49d.r8
32: (t1) >@d1rg>@u41'a<dfa',2t61@u37@d0'a+<dfa+',2
33: (t1) @d1<@u47dt57r>@d0@d1t54@u41'ea+<c+ea',3t47r@d0
34: (t1) t37@u40>'f<cga<cf'144,3r
35:
36: (t2) r32@5@b0@v105y11,127r3214@q1p3
37: (t2) |:@u49o4|:4'fga':| @u31'fga+''fga+'@u49'ega+'@u56'ega+'
38: (t2) @u51'd+ga+'@u56'd+ga+'@u62'd+f+a<d'@u66'd+f+a<d'
39: (t2) @u63'df<c'@u58'df<c'@u54'dfa+''dfa'
40: (t2) @u57'c+fga+''c+fga+'@u53'c+fga+'@u49'c+fga+'
41: (t2) 'cega''cega''cefa''cefa'|>@u56'b<df''b<df''b<df''b<df'<
42: (t2) @u53'cfg'@u49'cfg'@u37'ceg'@u26t57'ceg':|
43: (t2) @u56>'b<df''b<df'@u56'b<de'@u64'b<de'
44: (t2) @u62'b<c+e'@u69'b<c+e'@u78@q1'b<c+e'@q2'b<c+e'
45: (t2) @q1@u79'aa+<df''aa+<df'@u70'aa+<df''aa+<df'
46: (t2) @u51'a<cef''a<cef'@u44'a<cef'@u38'a<cef'
47: (t2) @u60'g+<cd+g''g+<cd+g'@u58'g+<cd+g'@u56'g+<cd+g'
48: (t2) @u49'a+<cd+''a+<cd+'@u52'a+<cd+''a+<cd+'
49: (t2) @u62'g+a+<cd+''g+a+<cd+'@u71'ga+<cd+'@u78'ga+<cd+'
50: (t2) @u79'fg+<cd'@u70'fg+<cd'@u63'fgb<d'@u53'fgb<d'
51: (t2) @u59<@q2'cdf''cdf'@u40@q3'cdf'@u34@q1'cdf'
52: (t2) @u34@q0'ceg''cega'@u31'ceg'@u17'ceg'@q1
53: /* 以下6行は | の位置以外上と同じ
54: (t2)|:@u49o4|:4'fga':|@u31'fga+''fga+'@u49'ega+'@u56'ega+'|
55: (t2) @u51'd+ga+'@u56'd+ga+'@u62'd+f+a<d'@u66'd+f+a<d'
56: (t2) @u63'df<c'@u58'df<c'@u54'dfa+''dfa'
57: (t2) @u57'c+fga+''c+fga+'@u53'c+fga+'@u49'c+fga+'
58: (t2) 'cega''cega''cefa''cefa' >@u56'b<df''b<df''b<df''b<df'<
59: (t2) @u53'cfg'@u49'cfg'@u37'ceg'@u26t57'ceg':|
60: (t2) @u51'd+ga+''d+ga+'@u59'd+f+a<d''d+f+a<d'
61: (t2) @u56'df<c'@u49'df<c'@u56'c+fga+''c+fga+'
62: (t2) 'cega'@u49@q1'cega'@u40'f+a+<d''f+a+<d'
63: (t2) @d1'faa+<d'r@d0r2 @d1'cfga+'>c@d0@d1rc@d0 r1
64:
65: (p)
```

リスト4 Planet of Lifeカウンタ表示

```
1:00001E0C 00000000   2:00001E0C 00000000
```

リスト5 スーパーマリオ(地上)

```
==================  HVC_SM01.NEW  ==================
  1: .comment -超・魔璃御兄弟- 地上 (C)Nintendo  by ENG 95/02
  2:
  3: / for ZMUSIC.X (v2.04以上推奨)
  4:
  5: /------------------------------------------------------------
  6: / TRACK SETUP
  7:
  8: (i)(d0)
  9:
 10: (m1,3000)(aFm1,1)
 11: (m2,3000)(aFm2,2)
 12: (m3,3000)(aFm3,3)
 13: (m4,3000)(aFm8,4)
 14: (m5,3000)(aFm4,5)
 15:
 16: /------------------------------------------------------------
 17: / OPM DATA
 18:
 19: /        AR DR SR RR SL  OL KS ML D1 D2 AM        BASS
 20: (@1      31  0  0 10  0  32  0 10  0  2  0
 21:          31  0  0 10  0  62  0  0  0  0  0
 22:          31  0  0 10  0  50  0  1  0  0  0
 23:          27  0  0 13  0   2  0  0  0  0  0
 24: /        AL FB
 25:           3  7)
 26:
 27: /        AR DR SR RR SL  OL KS ML D1 D2 AM        SQU
 28: (@2      31  0  0 15  0  25  0  2  0  0  0
 29:          31 29  0 15  2  35  0  3  0  0  0
 30:          31 29  0 15  2  35  0  5  0  0  0
 31:          31 29  0 15  2   0  0  1  0  0  0
 32: /        AL FB
 33:           5  7)
 34:
 35: /        AR DR SR RR SL  OL KS ML D1 D2 AM        NOIZ
 36: (@10     31  0  0 15  0 127  0  0  0  0  0
 37:          31  0  0 15  0 127  0  0  0  0  0
 38:          31  0  0 15  0 127  0  0  0  0  0
 39:          25 31 14 13  5   0  0  0  0  0  0
 40: /        AL FB
 41:           0  0)
 42:
 43: /------------------------------------------------------------
 44: / WAVE DATA
 45:
 46: .wave_form 8,0  { 1   0   0   0   0  -2  -2  -2
 47:                  -2  -4  -4  -4  -4  -6  -6  -6
 48:                  -6  -8  -8 -80 }
 49:
 50: /------------------------------------------------------------
 51: / MML DATA
 52:
 53: (t1)    @1@v120@k772
 54:         L8o3 @q4
 55:         ddrdrddr <@k760gr4.@k772>gr4.                        /1
 56:         [do]
 57:         |:
 58:         |:gr4er4cr4frgrg-fr@q12
 59:         {e<c@k762e}2@q4@k758fr@k769d@k762er@k772cr>abgr4:|
 60:         ||:
 61:         |:cr4gr4<cr>fr4<ccr>fr|
 62:         cr4er4g<cr@k760_1<grggr>>@k772~gr:|
 63:         cra-r4b-r4<cr4>ggrcr:|
 64:         >|:3a-r4<e-r4a-rgr4cr4>gr:|<                         /2
 65:         ddrdrddr <@k760gr4.@k772>gr4.                        /1
 66:         :||:
 67:         |:cr4f+gr<cr> frfr<cc>fr|                            /3
 68:         dr4fgrbr grgr<cc>gr:|gr4g@q12{gab}2@q4<cr>grcr4.:|
 69:         >|:3a-r4<e-r4a-rgr4cr4>gr:|<                         /2
 70:         ddrdrddr <@k760gr4.@k772>gr4.                        /1
 71:         |:cr4f+gr<cr> frfr<cc>fr|                            /3
 72:         dr4fgrbr grgr<cc>gr:|gr4g@q12{gab}2@q4<cr>grcr4.
 73:         [loop]
 74:
 75: (t2,3)  r*1 @2 @v123
 76:         s,8 @s,1 @h,0 @a1
 77:         L8o5 q8
 78: (t2)    ee4e4ce4g2>g2                                        /1
 79: (t3)    >f+f+4f+4f+f+4b1                                     /
 80: (t2,3)  [do]|:
 81: (t2)    |:<c4.>g4.e4.a4b4b-a4 {g<eg}2a4fg4e4cd>b4.:|<
 82: (t3)    |:e4.c4.>g4.<c4d4d-c4 {cgb}2<c4>ab4a4efd4.:|
 83: (t2,3)  |
 84: (t2)    |:|:r4gg-fe-4e4> a-a<c4>a<cd r4
 85:         |      gg-fe-4e4 <c4cc2>:|e-4.d4.c1:|
 86: (t3)    |:|:r4<ee-dc-4c4> e f g4 c ef r4
 87:         |      <ee-dc-4c4 f4ff2>:|a-4.f4.e1:|
 88: (t2)    |:c c 4c 4c d 4ec4>ag2|<c c 4c 4c d e*216:|          /2
 89: (t3)    |:a-a-4a-4a-b-4ge4 ec2| a-a-4a-4a-b-g*216:|          /
 90: (t2)    ee4e4ce4g2>g2<                                       /1
 91: (t3)    f+f+4f+4f+f+4b1                                      /
 92: (t2,3)  :||:
 93: (t2)    |:ec4>g4.g+4a<f4f>a2                                 /3
 94:         |{b<aaagf}1ec4>ag2:|b<f4f{fed}2c1:|                  /
 95: (t3)    |:<c>a4e4.e4f<c4c>f2                                 /
 96:         |{g<fffed}1c>a4fe2:|g<d4d{dc>b}2ge4ec2:|             /
 97: (t2)    |:c c 4c 4c d 4ec4>ag2|<c c 4c 4c d e*216:|          /2
 98: (t3)    |:a-a-4a-4a-b-4ge4 ec2| a-a-4a-4a-b-g*216:|          /
 99: (t2)    ee4e4ce4g2>g2<                                       /1
100: (t3)    f+f+4f+4f+f+4b1                                      /
101: (t2)    |:ec4>g4.g+4a<f4f>a2                                 /3
102:         |{b<aaagf}1ec4>ag2:|b<f4f{fed}2c1                    /
103: (t3)    |:<c>a4e4.e4f<c4c>f2                                 /
104:         |{g<fffed}1c>a4fe2:|g<d4d{dc>b}2ge4ec2               /
105: (t2,3)  [loop]
106:
107: (t4)    @10 @v127
108:         @q22 o6 L8
109:         @o31
110:         |:3c4,*13|c:|c4.,*13c4,*13ccc                        /1
111:         [do]
112:         |:24@o0o3c4,*2o6@o31c*30,*2c*18,*2                   /2
113:               c4,*13c*30,*2c*18,*2:|                         /
114:         |:|:4
115:         |:3c4,*13|c:|c4.,*13c4,*13ccc:|                      /1
116:         |
117:         |:8 @o0o3c4,*2o6@o31c*30,*2c*18,*2                   /2
```


▶うっ，6月号の特別企画は……。HDDは同じやり方で内蔵(HDDなしモデル)したし，プリンタはBJC-400Jだし，DrawingSlateももってるし……。ああっ，Oh!Xに洗脳されているんだろうか。
遠藤　勝博(25)宮城県

```
118:                  c4,*13c*30,*2c*18,*2:|                       /
119:           |:16 c*75,*2c*21,*2 c4,*13c4,*2:|                   /3
120:           :|
121:           |:8  c*75,*2c*21,*2 c4,*13c4,*2:|                   /3
122:           [loop]
123:
124: (t5)      [do]
125:           @t231r*3@t232r*5
126:           [loop]
127:
128: (p)
```

リスト6　スーパーマリオ(水中)

```
=================  HVC_SM03.ZMS  =================
  1: .comment -超・魔璃御兄弟- 水中 (C)Nintendo  by ENG 95/02
  2: .
  3: / for ZMUSIC.X (v2.04以上推奨)
  4:
  5: /----------------------------------------------------------
  6: / TRACK SETUP
  7:
  8: (i)(d0)
  9:
 10: (m1,3000)(aFm1,1)
 11: (m2,3000)(aFm2,2)
 12: (m3,3000)(aFm3,3)
 13: (m4,3000)(aFm8,4)
 14: (m5,3000)(aFm4,5)
 15:
 16: /----------------------------------------------------------
 17: / OPM DATA
 18:
 19: /         AR DR SR RR SL  OL KS ML D1 D2 AM          BASS
 20: (@1       31  0  0 10  0  32  0 10  0  2  0
 21:           31  0  0 10  0  62  0  0  0  0  0
 22:           31  0  0 10  0  50  0  1  0  0  0
 23:           27  0  0 13  0   2  0  0  0  0  0
 24: /         AL FB
 25:            3  7)
 26:
 27: /         AR DR SR RR SL  OL KS ML D1 D2 AM          SQU
 28: (@2       31  0  0 15  0  25  0  2  0  0  0
 29:           31  0  0 15  0  43  0  3  0  0  0
 30:           31  0  0 15  0  43  0  5  0  0  0
 31:           31  0  0 15  0   8  0  1  0  0  0
 32: /         AL FB
 33:            5  7)
 34:
 35: /         AR DR SR RR SL  OL KS ML D1 D2 AM          PSG2'
 36: (@3       31  0  0 15  0  19  0  1  0  0  0
 37:           31  0  0 15  0  36  0  1  0  0  0
 38:           31  0  0 15  0  28  0  1  0  0  0
 39:           27  0  0 15  0   9  0  3  0  0  0
 40: /         AL FB
 41:            3  5)
 42:
 43: /         AR DR SR RR SL  OL KS ML D1 D2 AM          NOIZ
 44: (@10      31  0  0 15  0 127  0  0  0  0  0
 45:           31  0  0 15  0 127  0  0  0  0  0
 46:           31  0  0 15  0 127  0  0  0  0  0
 47:           25 31 14 13  5   0  0  0  0  0  0
 48: /         AL FB
 49:            0  0)
 50:
 51: /----------------------------------------------------------
 52: / WAVE DATA
 53:
 54: .wave_form 8,0 { -7  -5  -2   0   0   0   0   0
 55:                   0   0   0   0   0   0   0   0
 56:                   0   0   0  -1  -3  -3  -5  -5
 57:                  -7  -9 -12 -16 -20 -26 -33 -41
 58:                 -52 -80}
 59:
 60: .wave_form 9,0 { -7  -4  -2   0   0 -80}
 61:
 62: /----------------------------------------------------------
 63: / MML DATA
 64:
 65: (t1)      r*4@1@v120@k772 L8o3
 66:           [do]
 67:           @q13 r1gg4.,4.
 68:           |:cg<c> c-gb cg<c> eg<c>:|
 69:           dgb d-g-b- dgb c-gb
 70:           dgb c-gb cg<c> >g<g<c>
 71:           cg<e> >b<g<d> >b-<g<d-> c+g<e>
 72:           da<f> d-a<f> ca<f> >b<g<f>
 73:           >c<g<e> >g<gg @q1{ff}f.>b16<c4.
 74:           [loop]
 75:
 76: / 記号 [`] は SHIFT+@
 77:
 78: (t2,3)    r*4@2
 79:           s,8 @s,2 @h,0 @a1 L*1o4
 80: (t2)      @v120[do]
 81:           @3d`@2d*23@3e`@2e*23@3f+`@2f+*23
 82:           @3g`@2g*23@3a`@2a*23@3b-`@2b-*23
 83:           |:@3b`@2b*11:|s,9|:@3b`@2b*23:|
 84:                s,8@3b`@2b*47@3g`@2g*23<
 85:           @3e`@2e*71@3e-`@2e-*71@3e`@2e*83>
 86:           @3g`@2g*11@3a`@2a*11
 87:                @3b`@2b*11<@3c`@2c*11@3d`@2d*11
 88:           @3e`@2e*71@3e-`@2e-*47@3f`@2f*23
 89:                @3e`@2e*131>@3g`@2g*11<
 90:           @3d`@2d*71@3d-`@2d-*71@3d`@2d*83>
 91:           @3g`@2g*11@3a`@2a*11
 92:                @3b`@2b*11<@3c`@2c*11@3d-`@2d-*11
 93:           @3d`@2d*71>@3g`@2g*47<@3f`@2f*23
 94:                @3e`@2e*131>@3g`@2g*11<
 95:           |:3@3g`@2g*71:|
 96:           @3g`@2g*23s,9@3a`@2a*35s,8@3g`@2g*11
 97:           |:3@3f`@2f*71:|
 98:           @3f`@2f*23s,9@3g`@2g*35s,8@3f`@2f*11
 99:           @3e`@2e*71>@3a`@2a*23@3b`@2b*23<@3f`@2f*23
100:           |:@3e`@2e*11:|@3e`@2e*35>@3b`@2b*11<@3c`@2c*71
101:           [loop]
102:
103: (t3)      @v120[do]
104:           @3d`@2d*23@3d-`@2d-*23@3c`@2c*23
105:           @3c-`@2c-*23@3c`@2c*23@3c+`@2c+*23
106:           |:@3d`@2d*11:|s,9@3d`@2d*23@3e`@2e*23s,8@3f`@2f*71
107:           @3g`@2g*71@3g-`@2g-*71@3g`@2g*143
108:           @3g`@2g*71@3g-`@2g-*47@3a`@2a*23@3g`@2g*143
109:           @3f`@2f*71@3f-`@2f-*71@3f`@2f*143
110:           @3f`@2f*71@3c-`@2c-*47@3a`@2a*23@3g`@2g*143<
111:           @3e`@2e*71@3d`@2d*71@3d-`@2d-*143
112:           @3d`@2d*71@3d-`@2d-*71@3c`@2c*143>
113:           @3c`@2c*71@3f`@2f*23@3g`@2g*23@3b`@2b*23
114:           |:@3b`@2b*11:|@3b`@2b*35@3f`@2f*11@3e`@2e*71
115:           [loop]
116:
117: (t4)      r*4 @10 @v125 @o31
118:           @q11 o6 L8
119:           [do] rc,*1c4,*7{cc}c8,*7 [loop]
120:
121: (t5)      [do] @t213r*2 @t212r*1 [loop]
122:
123: (p)
```

リスト7　スーパーマリオ(ゲームオーバー)

```
=================  HVC_SM06.ZMS  =================
  1: .comment -超・魔璃御兄弟- 終了 (C)Nintendo  by ENG 95/02
  2:
  3: / for ZMUSIC.X (v2.04以上推奨)
  4:
  5: /----------------------------------------------------------
  6: / TRACK SETUP
  7:
  8: (i)(d0)
  9:
 10: (m1,3000)(aFm1,1)
 11: (m2,3000)(aFm2,2)
 12: (m3,3000)(aFm3,3)
 13: (m4,3000)(aFm4,4)
 14: (m5,3000)(aFm8,5)
 15: (m6,3000)(aFm5,6)
 16:
 17: /----------------------------------------------------------
 18: / OPM DATA
 19:
 20: /         AR DR SR RR SL  OL KS ML D1 D2 AM          BASS
 21: (@1       31  0  0 10  0  32  0 10  0  2  0
 22:           31  0  0 10  0  62  0  0  0  0  0
 23:           31  0  0 10  0  50  0  1  0  0  0
 24:           27  0  0 13  0   2  0  0  0  0  0
 25: /         AL FB
 26:            3  7)
 27:
 28: /         AR DR SR RR SL  OL KS ML D1 D2 AM          SQU
 29: (@2       31  0  0 15  0  25  0  2  0  0  0
 30:           31  0  0 15  0  39  0  3  0  0  0
 31:           31  0  0 15  0  39  0  5  0  0  0
 32:           31  0  0 15  0   4  0  1  0  0  0
 33: /         AL FB
 34:            5  7)
 35:
 36: /         AR DR SR RR SL  OL KS ML D1 D2 AM          PSG2'
 37: (@3       31  0  0 15  0  19  0  1  0  0  0
 38:           31  0  0 15  0  36  0  1  0  0  0
 39:           31  0  0 15  0  28  0  1  0  0  0
 40:           27  0  0 15  0   6  0  3  0  0  0
 41: /         AL FB
 42:            3  5)
 43:
```

▶西川さん，あれをラップだと思うからいけないんですよ。あれは○○音頭っすよ，音頭。
高橋　毅(23)埼玉県

```
 44: /          AR DR SR RR SL  OL KS ML D1 D2 AM         PSG2
 45: (@4        31  0  0 15  0  25  0  2  0  0  0
 46:            31  0  0 15  0 127  0  2  0  0  0
 47:            31  0  0 15  0 127  0  1  0  0  0
 48:            31  0 12 15  0   0  0  1  0  0  0
 49: /          AL FB
 50:             2  7)
 51:
 52: /          AR DR SR RR SL  OL KS ML D1 D2 AM         NOIZ
 53: (@10       31  0  0 15  0 127  0  0  0  0  0
 54:            31  0  0 15  0 127  0  0  0  0  0
 55:            31  0  0 15  0 127  0  0  0  0  0
 56:            25 31 14 13  7   0  0  0  0  0  0
 57: /          AL FB
 58:             0  0)
 59:
 60: /----------------------------------------------------------
 61: / WAVE DATA
 62:
 63: .wave_form 8,0{    2   0   0  -1  -2  -4  -5 -10 }
 64:
 65: .wave_form 9,0{    0  -2  -4  -6  -8 -10 -13 -16
 66:                  -20 -25 -80 }
 67:
 68: .wave_form10,0{  -32 -64 -96 -128 -160 -192 -224 -256}
 69:
 70: .wave_form11,0{    0   0   0   0  -3  -3  -3  -3
 71:                   -6  -6  -6  -6  -9  -9  -9  -9
 72:                  -12 -12 -12 -12 -15 -15 -18 -18
 73:                  -21 -24 -27 -33 -44 -64 -80}
 74:
 75: .wave_form12,0{   0  -2  -4  -7 -11 -16 -21 -28
 76:                 -37 -80 }
 77:
 78: .wave_form13,0{  -4  -4  -2  -1   0   0   0   0
 79:                   0   0   0   0   0   0   0   0
 80:                   0   0   0   0  -1  -2  -4  -5
 81:                  -7  -8 -10 -11 -13 -15 -18 -21
 82:                 -24 -27 -30 -33 -37 -44 -60 -80}
 83:
 84: .wave_form14,0{  -4  -2  -1   0   0   0 -80}
 85:
 86: .wave_form15,0{   1  -2  -2  -2  -2  -2  -4  -4
 87:                  -4  -4  -4  -6  -6  -6  -6  -8
 88:                  -8  -8 -10 -10 -80}
 89:
 90: /----------------------------------------------------------
 91: / MML DATA
 92:
 93: (t1)      @1@v120 @k772
 94:           L8o3 @q9
 95:           |:3 |:drar16<dr8.>a<d>@k644:|@k772 :|drar16<dr8.
 96:           L16o3@q1
 97:           o3b<b8,*12b8.,*30c<c8,*12c8.,*30>
 98:           c+<c+8,*12c+8.,*30>>g8,*12g2
 99:           @v0r2@v120
100:           L8o3@q4grrg@q12{gab}2@q4<cr>grcr*146
101:           g8.,*10e8.,*10c8&f4&d-*81&c1,*70
102:
103: (t2)      r*1 @2 @v118 @k0
104:           s,15 @s,1 @h,0 @a1
105:           L8 @q8 o5
106:           |:4
107:           c c c. ,*16c|c. ,*16c c
108:           c-c-c-.,*16c-c-.,*16c-c-
109:           :|c*35,*16
110:
111: (t3)      r*1 @2 @v118
112:           s,15 @s,1 @h,0 @a1
113:           L8 @q8 o4
114:           |:4
115:           fff,*14_1d16,*11~ff,*14_1|d16,*11~f16,16_d16,*11~f
116:           eee,*14_1c16,*11~ee,*14_1 c16,*11~e16,16_c16,*11~e
117:           :|d*11,*10
118:
119: (t2)      @v116 s,14@s,2@h,0@a1
120:           L*1q8o4
121:           |:3
122:           o4@3e*2`@2e*10o5@3d*1`@2d*23@3d*2@2d*10@3d*1`@2d*23
123:           |1@k64:||2@k128:||3@k192:|
124:           @3d*1`@2d*23s,13@3d*1`@2d*15&@s,4d*45&@s,3d*35@k0
125:           @v0 L24o4
126:           @t235 @4@v123 s10,8@s1,1@m1
127:           @k20@h2,0b-b-b-
128:           _2 y$21,$e8 y$61,18y$71,23y$69,32 y$41,1
129:           s,9@k-20@h0b-*10@v0 @t231 L8
130:           r*62 @v123 @k-64
131:           s3,11@s16,1@h1,2@m-800
132:           @2g<d4,*32d{dc>b}2ge4,*32ec2,*32
133:           r*74 @t209 L8
134:           @m@s,2@h,0 @v116@k0o5 s,14 L*1
135:           @3c`@2c*35>@3g`@2g*35s,13@3e`@2e*23
136:           @3a`@2a*15@3b`@2b*15@3a`@2a*15
137:           @3a-`@2a-*26@3b-`@2b-*26@3a-`@2a-*26
138:           @3g`@2g*191,*70
139:
140: (t3)      @v116 L16o4q8
141:           s,13 @s,2 @h,0 @a1
142:           o3e<a-8a-a-8>f<a8aa8>f+<b-8b-b-8s,14
143:           @3b*1`@2b*23s,13@3b*1`@2b*15&@s,4b*45&@s,3b*35
144:           @v0 r2 @v124 L8
145:           s,12 @s,4 o4b<f4f{fed}2c1
146:           r*74 @s,2@v116o4 s,14 L*1
147:           @3e`@2e*35@3c`@2c*35s,13>@3g`@2g*23<
148:           @3f`@2f*47@3f`@2f*80
149:           @3e`@2e*11@3d`@2d*11@3e`@2e*36&@s,1e*131,*10
150:
151: (t4)      @v0 |:204 @t223r*2@t224r*2@t223r*2@t224r*2 :| @t235
152:
153: (t5)      @10 @v127 o6 L8
154:           |:14
155:           @o0o3c8,*1o6@o31c*16,*1c*8,*1|c8,*9c*16,*1c*8,*1
156:           :|@v0@o
157:
158: (p)
```

リスト8　スーパーマリオ(地上)カウンタ表示

```
==================  HVC_SM01.CO  ==================
  External Internal    External Internal    External Internal    External Int
ernal
1:00000180 00003600  2:00000181 00003600  3:00000181 00003600  4:00000180 000
03600
5:00000000 00000008
```

リスト9　スーパーマリオ(水中)カウンタ表示

```
==================  HVC_SM03.CO  ==================
  External Internal    External Internal    External Internal    External Int
ernal
1:00000004 00000900  2:00000004 00000900  3:00000004 00000900  4:00000004 000
00090
5:00000000 00000003
```

リスト10　スーパーマリオ(ゲームオーバー)カウンタ表示

```
==================  HVC_SM06.CO  ==================
  External Internal    External Internal    External Internal    External Int
ernal
1:00000A2B 00000000  2:00000A2B 00000000  3:00000A2B 00000000  4:00000660 000
00000
5:00000510 00000000
```

(善)のゲームミュージックでバビンチョ

西川善司

●レイフォース
VHS：PCVP-11662　4,800円(税込み)
LD：PCLP-00554　4,800円(税込み)
ポニーキャニオン　発売中

格闘全盛の時代にシューティングゲームで健闘している「レイフォース」。一見なんの変哲もない平面縦スクロールものなのだが，タイトーが誇る２次元画像処理技術の粋を結集して作り出される視覚的演出がゲーム展開を実に立体的に引き立てる。

ビデオはオープニングデモから最終ステージクリア/エンディングまでのすべてのシーンを収録。ただし攻略色はきわめて薄く，ゲームビジュアルを絵巻のように楽しむことをコンセプトにした構成といえよう。

今回，演出にこだわるZUNTATAはこのビデオを企画制作の段階からプロデュース。単にゲーム映像を収録したのではなく，ゲームの設定に基づいたドラマチックな演出がなされている。謎の惑星に不時着したレイフォースの自機戦闘機のパイロットである女性生体兵器(橋本典子)はシステムダウン寸前に生前の記憶を再生する……といった感じ(収録時間55分)。

・おすすめ度　8

●餓狼伝説3-ARRANGE SOUND TRAX-SNK　新世界楽曲雑技団
CD：PCCB-00183　2,500円(税込み)
ポニーキャニオン　発売中

いわゆるアレンジバージョンアルバムなのだが，従来のモノとは少し違う。最近のNEO･GEOゲームはゲームセンター(ROM)版と家庭用NEO･GEO CD版とでBGMが異なっており，NEO･GEO CD版ではその大容量性を活かした新収録のアレンジバージョンが演奏されるのだ。今回のアルバムはそのNEO･GEO CD版のBGMを全収録したもの。エンディング曲はボーカルバージョンを不知火舞役の声優が歌っている。

・おすすめ度　7

●ナムコゲームサウンドエクスプレス Vol.20　アウトフォクシーズ
CD：VICL-15042　1,500円(税込み)
ビクターエンタテインメント　6/21発売

スパイに扮し，ステージ中にあるすべてのものを駆使して相手を倒すちょっと変わった対戦アクションゲーム。「殺し屋」「スパイ」モノというと古くからのナムコファンならば「ローリングサンダー」シリーズを思い浮かべるだろう。BGMは作曲者こそ違うもののノリ自体はかなり「ローリングサンダーII」に近いものがある。コンガがトコタカとアクセントをつけたリズムに乗せて，サックス，ピアノが即興性の高い旋律を展開する。そしてココというときにバックブラス隊がリピジョンで被る……といった期待どおりのサウンドがうれしい。ところで全チャンネルPCM音源の演奏なんだろうけどちょっと音が薄い気がするのが気になった。

・おすすめ度　8

●パーフェクトセレクション ドラキュラバトルII
CD：KICA-1162　3,000円(税込み)
キングレコード　6/21発売

前作「ドラキュラバトルI」から約1年，またまたヘビメタ/ハードロックアレンジで「ドラキュラバトルII」のタイトルを引っ提げて再登場。単にギターで演奏したグレードアップバージョンでなく，曲自体にかなり手を入れ本格的なギターサウンドに仕上がっている。とはいえ，ちゃんと原曲の持ち味は失われておらずオリジナルファンの期待も裏切ってはいない。

曲目はお馴染みの「Beginning」などからX68000版「悪魔城ドラキュラ」専用曲「Theme of Simon」「Thrashard in the Cave」なども収録。ドラキュラファンはまた必携の１枚が増えたってことかな。

・おすすめ度　9

●ぽっぷるメイル　パラダイス　3
CD：KICA-11161　2,800円(税込み)
キングレコード　6/21発売

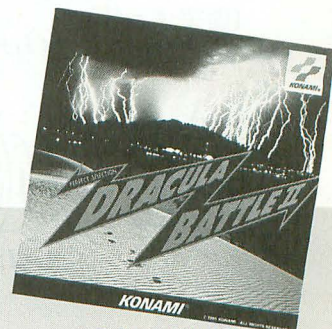
現在TBSラジオ系(毎週土曜AM２：00より)で放送されているファルコムレーベル情報番組「TARAKOファルコムぴーヒャララ」。この番組の人気コーナーであるラジオドラマをCDに収録したもの。タイトルはお馴染みコミカルアクションRPG「ぽっぷるメイル」。このシリーズも早くも3作目となった。回を増すごとにキャラクターの役どころが明確になってきているためか，わかりやすい笑いがストーリーの至るところに散りばめられている。声優ファンのコレクターズアイテムの色が濃かったこの系統のドラマもだんだんとシナリオの質が高くなってきた。面白い。もしかしたら今後，こういったラジオドラマが原作のゲームができるみたいな本末転倒的進化もありうるか。なお，番組では今後ファルコムゲームを続々ドラマ化していく方針とか。

・おすすめ度　7

●もっと！　ときめきメモリアル JUN. ～featuring 鏡魅羅～
CD：KICA7662　2,800円(税込み)
キングレコード　6/21発売

プレイステーション，セガサターンにも移植の決定したコナミの学園ドラマゲーム「ときめきメモリアル」。なんかいままでのコナミのイメージとは違ったゲームだが，世間の評価は高いようでファンも多い。さて，この「ときめきメモリアル」は文化放送(毎週日曜日24:00)，東海ラジオ(毎週火曜日22：00)，山梨(毎週木曜22：30)にてラジオドラマとして放送されており，今回のCDはこれを収録したもの。CD後半には留守電やエラービープ音に最適な台詞集が収録されている。

・おすすめ度　7

レイフォース

ドラキュラバトルII

SIDE A

ゼロヨンゲーム完結……一応(後編)

Tan Akihiko 丹 明彦

今月は前回解決しなかったホイールスピンの挙動も解決し
これで直線での車の挙動がだいたい把握できた
次回からは，いよいよコーナリングまで含めた車の挙動に迫ってみる

前回未解決だった「スタート時の挙動」だが，一応説明できる程度に完成した。前回はこのへんを言葉でのみ説明していたが，今回は式を交えて解説する。プログラムも妙な係数やパラメータを導入することなく動作しているが，ソースプログラムを誌面に掲載できる形で整えるのは時間的な都合で断念した。

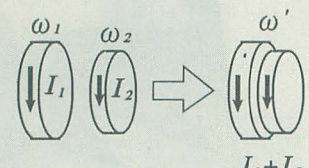
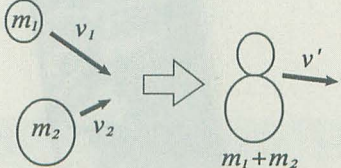
前回の内容の訂正

その前に6月号の内容で1カ所明らかな間違いに気づいたので訂正しておく。サスペンションの動作を説明した図6中の「速度変化」の式の最後にΔtをつけ忘れていた。正しくは，

$$\Delta v = \frac{Fin + Fout}{m}\Delta t$$

である。

しかし，式は間違っていたがプログラムは間違っていないので，サスペンションの挙動は正しくシミュレートされている。

回転運動に関する基礎知識

今回の内容を理解するためには，剛体の回転運動に関してある程度の知識が必要だ。確か高校までの物理では出てこない内容なので，簡単に紹介しておく。ここを理解するためには，高校1年で習う力学の基礎的な概念や方程式を理解しておくことが必要だろう。

ここで説明しようとしている回転運動は「自転」運動，つまり物体自身が回転する運動である。遠心力などの概念が出てくる回転運動は高校で習うが，これはいわば「公転」運動，つまり物体がある点を中心として円軌道を描く運動である。両者は区別しておく必要がある。

さて，回転運動を記述するためにさまざまな物理量や式が存在しているわけだが，これが都合のいいことに，一般の運動(平行移動)と概念が対応するのである。図1に並べてあるので理解の助けにしていただきたい。

図1 回転運動に関する基礎知識

回転運動		(参考)一般の運動	
慣性モーメント	I	質量	m
角速度	ω	速度	v
角運動量	$L=I\omega$	運動量	$p=mv$
トルク	T	力	F
回転運動方程式	$T=\frac{dL}{dt}=I\frac{d\omega}{dt}$	運動方程式	$F=\frac{dp}{dt}=m\frac{dv}{dt}$
角運動量保存則	$I_1\omega_1+I_2\omega_2=I_1\omega_1'+I_2\omega_2'$	運動量保存則	$m_1v_1+m_2v_2=m_1v_1'+m_2v_2'$
(衝突後一体)	$I_1\omega_1+I_2\omega_2=(I_1+I_2)\omega'$	(衝突後一体)	$m_1v_1+m_2v_2=(m_1+m_2)v'$

慣性モーメントはこれまでも何度か紹介したが，物体の回りやすさを示す量である。物体が単位時間にどのくらい回転しているかを表す量が角速度，物体の回転の状

態を変えるための力に相当する量がトルクである。トルクと慣性モーメントと角速度の間には，ちょうど運動方程式に相当する関係が成立している。

角運動量は慣性モーメントと角速度をかけた量。回転の運動方程式は，「トルク＝角運動量の時間微分」という関係を表したもので，慣性モーメントを定数として分離して記述した式をよく使う。ここでもその式だけを用いる。なお蛇足ながら，一般に，

F＝m・a

として認知されている運動方程式も，

力＝運動量の時間微分

と記述でき，必ずしも質量が一定でない相対性理論の世界ではこちらの式のみが成立する。

そして前回でも言葉だけは出しておいた角運動量保存則。いうまでもなく運動量保存則に対応する概念で，複数の回転する物体が衝突したあとの回転状態を得るための法則。今回用いるのは2つの回転する物体が接触後に一体となる場合である。

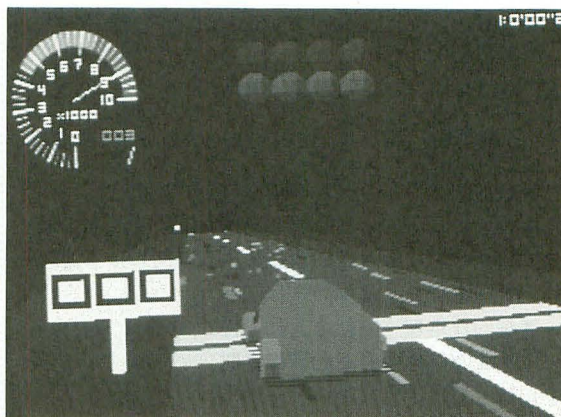
スタート時の挙動（定量的バージョン）

車が空ふかしからクラッチミート，ホイールスピンを経て走行を始めるまでの一連の挙動を記述するために，次のいくつかの量を導入する（図2）。

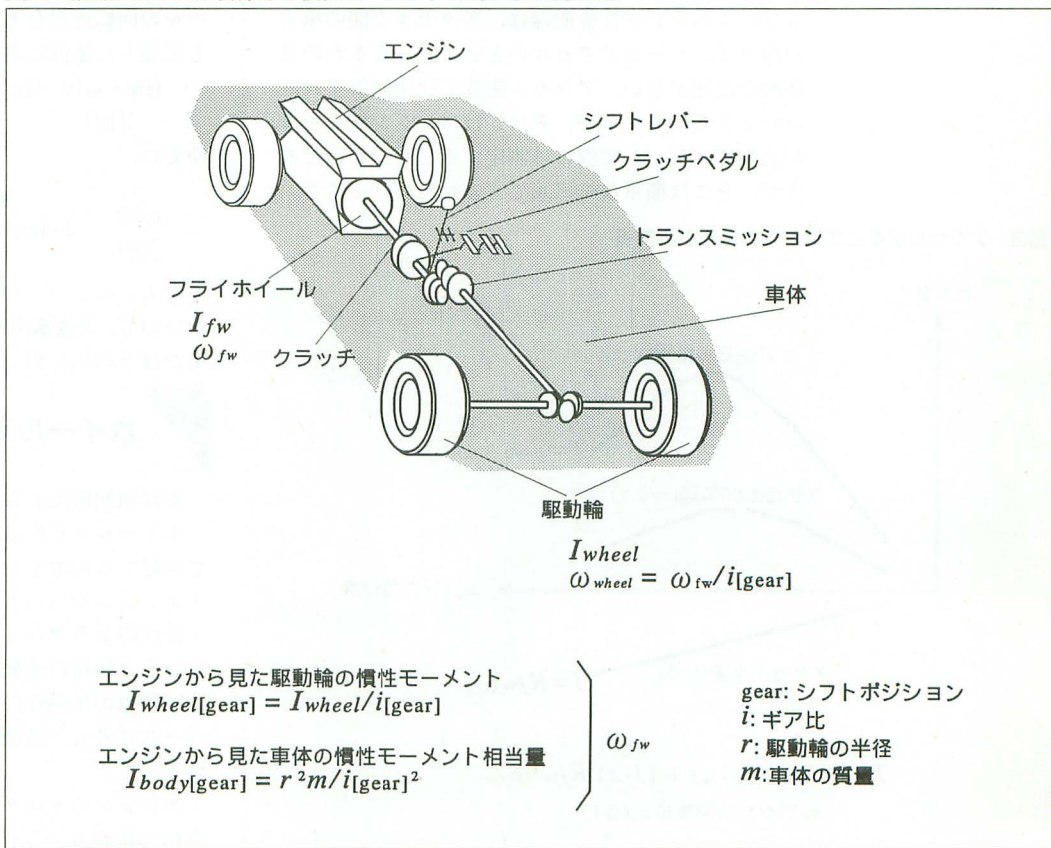
・エンジンのフライホイールの慣性モーメント

回転数エンジンの吹け上がりのよしあしや，負荷がかかったときの粘りに端的に反映する。たとえば軽いフライホイールを備えたエンジンは軽快に吹け上がるが，負荷に対する粘りには欠ける。エンジンのフライホイールの角速度をrpm単位に換算した量はエンジン回転数と呼ばれている。

・エンジンから見た駆動輪の慣性モーメント

ホイールスピン中の駆動輪の角速度の変化の度合に反映する。ここで仮想的に，「エンジンから見た」駆動輪の慣性モーメントを導入する。そうするとホイールスピン中のエンジン回転数の変化を直接知ることができる。エンジンと駆動輪はトランスミッションを介して接続されており，当然ながらこの値はシフトポジションによって異なる。たとえば，低いギアほどギア比は大きく，エンジンの立場から見れば軽く回るということになる。

・エンジンから見た車体の慣性モーメント相当量

これを導入したのは我ながら大胆だと思う。エンジン出力が車体を推進させ，結果としてエンジン回転数の増加に反映する。これをエンジンの立場から見れば，トランスミッションの先に仮想的な回転体を接続している状態と考えることも可能であろう。この値も当然，シフトポジションによって異なる。導入過程は省略するが，過去に解説した駆動力の式，

図2　駆動システムの各部分の慣性モーメントと対応する角速度

$$m\frac{dv}{dt}=\frac{T\cdot i[gear]}{r}$$

と，速度とエンジン角速度の関係式，

$$v=\frac{\omega fw\cdot r}{i[gear]}$$

を解けば図2に示した式が得られるであろう。

それでは具体的に各状態を記述していく。ある連続的状態から次の連続的状態に移る瞬間には不連続的な処理を行う。基本的には，

・連続的な処理→回転の運動方程式
・不連続な処理→角運動量保存則

を用いると考えればよい。

空ふかし（連続的挙動）

非常に単純で，フライホイールとエンジンのトルクTの関係にすぎない。

$$Ifw\frac{d\omega fw}{dt}=T$$

ハーフアクセルについて

ここで脱線して，懸案だった「アクセル開度とエンジンのトルクの関係」についてひとつのモデルを立てたので紹介しておく(図3)。車のカタログに載っているエンジン性能曲線は，アクセル全開の場合の出力で，ハーフアクセルのときにどうなるかの具体的な記述がない。アクセル開度で比例制御すればいいような気もするが，それではエンジンブレーキが表現できない。実際には測定するしかないのだろうが，そこは簡単なモデルですませたい。そこで，「エンジンの内部抵抗」を導入してみた。これはエンジン回転数に比例して増大する量で，アクセル全閉の場合はこれがエンジン出力となり，制動力のもととして働く。つまりエンジンブレーキである。ハーフアクセルの場合は，アクセル開度に応じて補間した値を用いる。

$$T = x\cdot Tmax(\omega fw)+(1-x)\cdot Rfw\cdot \omega fw$$

x:アクセル開度(0~1)
Tmax(ωfw):アクセル全開時のトルク
Rfw:エンジン内部抵抗係数

これが物理的にでたらめかといえば必ずしもそうではなく，

「出力曲線はエンジンの純粋な出力から内部抵抗を差し引いた値のグラフ」
「エンジンの純粋な出力はアクセル開度に比例する」
「エンジン内部抵抗はアクセル開度に依存しない」

などと仮定すればあながち的外れでもないだろう。

アクセル開度を得るためにサイバースティックの入力を試しに使ってみた。が，レバーだとなかなか微妙なアクセルワークができない。重めのペダルユニットは必要だろう。

図3　アクセル開度とエンジンのトルクの関係

クラッチミート（不連続的挙動）

今回のモデルではクラッチミートを，「フライホイールの回転運動をそっくり駆動輪に受け継ぐ」ことと定義し，素直に角運動量保存則を適用する。

$$Ifw\cdot \omega fw=Iwheel\cdot \omega wheel$$

[旧]　　　　[新]

ゆえに，

$$\omega fw=\frac{Ifw}{Iwheel[speed]}\omega fw$$

[新]　　　　[旧]

となる。エンジンの角速度(Ifw)をベースに計算したいので，駆動輪の慣性モーメントはエンジンから見たほうの値を用いる。

ホイールスピン（連続的挙動）

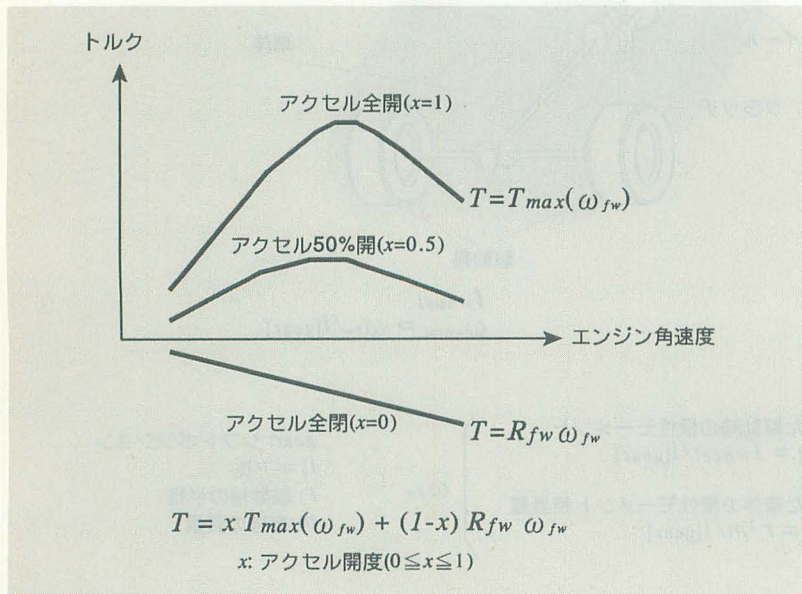
実は前回解決しなかったのがこの部分。

ホイールスピンは，タイヤが路面に接触した状態で空転する現象であり，

・エンジンからのトルクによる回転数の増加
・路面のトラクションによる回転の損失

という2種類の運動の混成である(図4)。

駆動輪の挙動なのでとりあえず駆動輪の角速度をベースに考え，最後にエンジン角速度に直すことにする。

エンジンのトルクはトランスミッションを通って車軸に伝わる。このとき回転数は1/i[gear]に減り

(このことからiは減速比と呼ばれる)，トルクはi[gear]倍に増大する(仕事の原理)。

$$Iwheel\frac{d\omega wheel}{dt}=i[gear]\cdot T$$

次に路面のトラクションだが，路面に力Fを与えて進んだ場合，駆動輪の回転を損失させる方向にF・rのトルクがかかる(rは駆動輪の半径)。

$$Iwheel\frac{d\omega wheel}{dt}=-F\cdot r$$

以上を総合してエンジン角速度の式に直すと，

$$Iwheel[gear]\frac{d\omega fw}{dt}=i[gear]\cdot T - F\cdot r$$

ということになる。

なお，路面に伝わる力Fは，駆動輪への荷重(もちろん荷重移動も考慮に入れる)をWとすると$F=\mu W$なのであるが，ホイールスピンしている間はμの値はグリップしているときよりも低下する(グリップ時の半分くらい)。

タイヤのグリップ回復(不連続的挙動)

これはタイヤがグリップした瞬間，それまで空転していた駆動輪の角運動量が車体の速度に上乗せされる現象である。

直線運動を表す量である車体速度に，回転運動を表す量である角速度を直接加算することはできないので，車体速度を仮想的なエンジン角速度に変換しておく。これをωbodyと名づける。むろんシフトポジションによって異なる。

$$\omega body[speed]=v\cdot i[gear]/r$$

角運動量保存則により，

$$Iwheel[gear]\cdot\omega fw[旧]+(Ibody[gear]-Ibody[gear])\cdot\omega body[gear]=Ibody[gear]\cdot\omega fw[新]$$

プログラム上は最終的に速度ベースに直すことにしたので次のようになる。

$$\underset{[新]}{v}=\underset{[旧]}{v}+\frac{Iwheel(\omega fw[旧] - \omega body[gear])}{r\cdot m}$$

グリップ状態での走行(連続的挙動)

以前にも解説したが参考までに。

$$m\frac{dv}{dt}=\frac{T\cdot i[gear]}{r}$$

ロケットスタートに関する考察

レースを見ていると，スタンディングスタートやピットアウトなどで激しくホイールスピンさせながら走り出していることがわかる。さきほど，ホイールスピン時はμの値が半分ほどに低下すると述べたが，これはつまり有効トラクションが減少することを意味している。しかし，スタート時に限ってはホイールスピンさせるくらいのほうが速いのである。これはなぜだろう。この問題が，今回立てたモデルをもとにある程度は説明できる。

グリップ走行をすると，低速域ではエンジンの低回転域を使わなくてはならない。そしてレース用にチューンされたレシプロエンジンの場合，低回転域でのトルクは高回転域でのトルクの半分もない。やはり有効トラクションをいっぱいに使って路面に力を伝えることはできないのである。

というわけなので，ある程度スピードが乗るまではホイールスピンさせ，エンジンを高回転域に保って駆動輪の回転を落とさず路面を蹴りながら加速したほうが速い場合もあるのである。

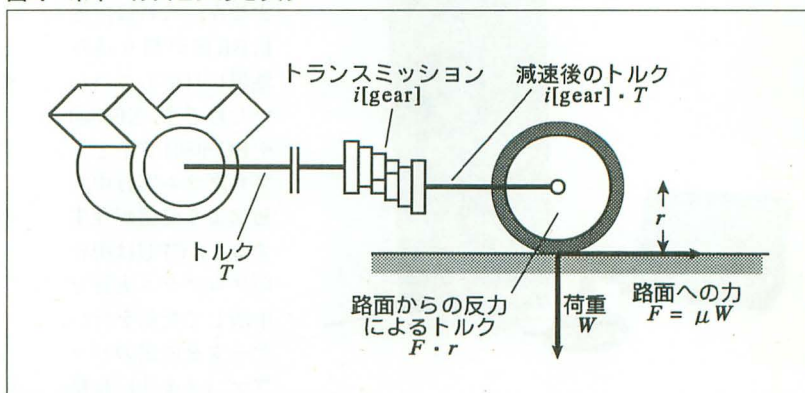
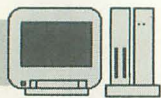
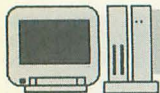
おわりに

正直に白状すると，今回は極めて「仮説」の色合が濃い。例によって，スタート時の挙動を明解に解説した自動車力学の本がない(少なくとも私は持っていない)ためである。

理論も根拠が薄いうえに，それを記述するためのパラメータ類についても当然ながら資料がない。車重とエンジンのトルクカーブとギア比はカタログを見ればわかるが，エンジンのフライホイールの慣性モーメントを知るための手がかりがない。

そういうわけなので，立てたモデルが正しいかどうかはわからないが，立てたモデルのシミュレーションに関しては正しく行われているはずである。

次はコーナリングまで含めた運動について考察を始めたいところだ。以前モデルを立てたが，どうやら半分くらいは嘘のような気がしている。あれ以来多少は経験を積んだので，今度はきちんとしたモデルを提示できると考えている。

図4 ホイールスピンのモデル

IBM PC/HP200LXとの接続実験

電机本舗 由井 清人 Yui Kiyoto

IBM PC/HP200LXとの接続実験を行います。多少の制限はあるとして，結局無事に接続ができたようです。また，いままで制作したもの「Z-Link」として誌上配布する告知があります。読み逃さないようにしましょう。

前回の最終回(!)でお約束したとおり，日本ヒューレットパッカード（旧名，横河ヒューレットパッカード）社のポケットパソコンHP200LXをX68000の仮想ドライブにする実験を行います。

まず，HP200LXを簡単に説明しましょう。このHP200LXは重量350g，乾電池で１カ月動作するパソコンです。デザインそのものはパソコンですし，事実IBM PC上位互換機です(写真１)。しかし，用途は，電子手帳と思ってください。背広のポケットにすっぽり入ります。そして，ROMに標準でロータス１２３，電話帳，予定表ソフトなどひととおりのものが組み込まれており，メニューよりワンタッチ選択できるようになっています。現在，電子手帳のシャープザウルスと，人気を二分している製品です。

戦略を練ろう

HP200LXはIBM PC互換機です。ゆえに，従機側のプログラムはIBM PC用に作ればよいことになります。

従機のプログラムはすべてＣ言語で記述していますので，X68000と共用できます。また，以前にPC-9801用にBIOSコールを利用し速度9,600bps限定で実験していますから，同様にIBM用を作ればよいわけです。

IBM PCの通信用BIOSは出来が悪く，事実上満足に動作しません。これは，通信BIOSが割り込み処理に対応していないためです。X68000やPC-9801ですと，プログラム実行中，通信より受信が発生すると，CPUは現在のプログラム実行を中断して受信を行い，データを所定のバッファ（メモリ）に格納します。ですから，プログラムから見ると，いつのまにか通信バッファにデータが格納されていることになります。

X68000とHP200LXの接続

対して，IBM PCはこのようなサービスがないため，受信したいときにはプログラムより，受信用のBIOSを呼んでいないといけません。事実上BIOSを使いポーリング処理をする形になります。速度の遅いBIOSコールを使い，さらにポーリング処理特有の受信バッファを使えないという二重苦をしょいこむ形になります。

このようなわけで，今回は，X68000版同様に通信LSIを直接制御しポーリングによる開発を行います。

i8250という石

IBM PCは，インテルのi8250という通信LSIを搭載しています。この石は，非同期専用のICです。X68000がもっているザイログZ8530が非同期に加え同期通信を備えているのと対照的に，機能を限定したLSIといえるでしょう。もっとも，かなり古いともいい直せますが……。

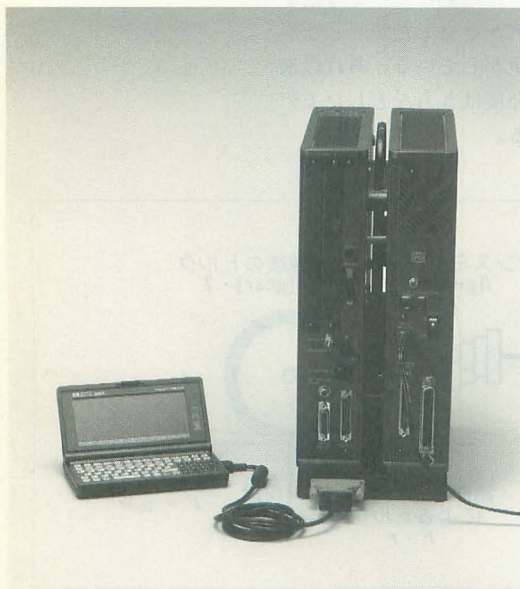
IBM-PCは伝統的にこの石を使用しています。これは最新のDOS/Vマシンにいたるまで変わりはありません。LSIの型式が異なっていても，上位互換品であり，質的に同じと断定してかまいません。

インテル系のCPUは，周辺LSIを制御するためのI/O空間をもっています。このI/O空間をポートアドレスと呼んでいます。このポートアドレスを利用してCPUと周辺LSIは連携動作を行います。モトローラ系は，通常メモリとポートアドレスの区別が同じ扱いになっており，このあたりの考え方が若干異なります。

図１に，i8250に割り振られているポートアドレスを示します。CPUはこのアドレスへデータを読み書きしてi8250をコントロールします。読み書きは，ポートアドレス制御用に専用のinとoutという命令が用意されています。

プログラムを紹介しましょう。まず，各ファイル共通の約束ごととして，

```
1：#define      X68      0
2：#define      PC-98    1
3：#define      IBM      2
4：#define      SYSTEM   1
```

以上のような記述があります。当プログラムはすべて，X68000，PC-9801，IBM PCの各機種共通になっています。

そして，機種依存のある行は，4行で定義している定数SYSTEMで判定し，コンパイルをするか否か決定しています。ですから，各機種でコンパイルするときにはSYSTEMに機種コードを設定してください。0＝X68000，1＝PC-9801，そして2＝IBM PCです。

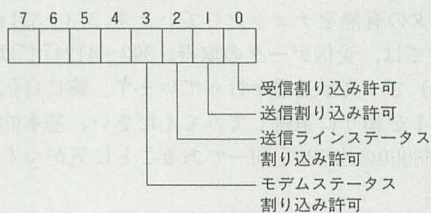
次に主機，従機共通のライブラリファイル“D3.C”を示します（リスト1）。前回までの“D3.C”と差し替えてください。このリストは，PC-9801およびIBM-PC用の通信LSIを直接制御する，高速通信制御プログラムが追加されています。従機プログラムをコンパイルリンクするときに利用してください。また，X68000機種依存の部分も高速化処理を加えてありますので，主機側プログラムもコンパイルリンクし直すといいでしょう。

具体的には通信データのブロック転送を行う関数を記述しています。特に基幹となるのは，送信ではblk_out1()という関数です。リスト中，次のように配置されています。

図1

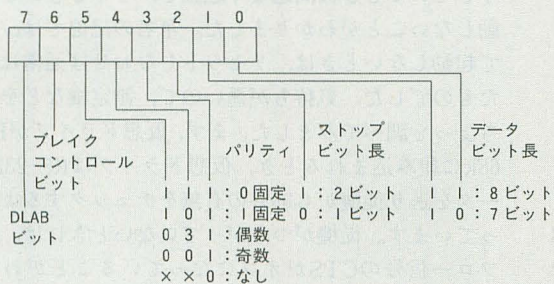
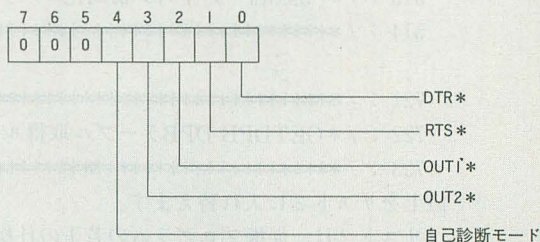
ポートアドレス	DLAB	1：入力 0：入力	レジスタ名	
3F8H	0	1	受信データレジスタ	
		0	送信ホールディングレジスタ	
	1	I/O	ボーレート分割レジスタLSB	
3F9H	0	I/O	割り込み許可レジスタ	（図1-1）
	1	I/O	ボーレート分割レジスタMSR	
3FAH		1	割り込み参照レジスタ	（図1-2）
3FBH		I/O	ラインコントロールレジスタ	（図1-3）
3FCH		I/O	モデムコントロールレジスタ	（図1-4）
3FDH		I/O	ラインステータスレジスタ	（図1-5）
3FEH		I/O	モデムステータスレジスタ	（図1-6）
3FFH		I/O	スクラッチパットレジスタ	

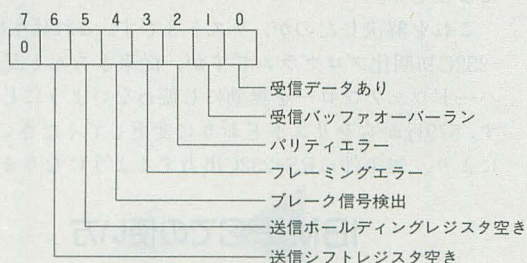
＊1：ポートアドレスはチャンネルが0が3Fxh，チャンネル1が2Fxh

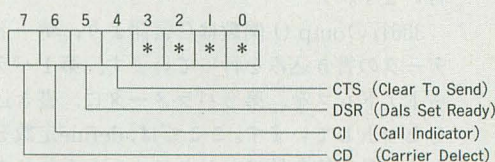
1-1　割り込み許可レジスタ

各ビットとも“1”で割り込みが許可となり，“0”で禁止となる

1-2　割り込み参照レジスタ

ビット210	割り込み優先度	割り込み種別	割り込みリセット	割り込み要因
110	高 ↑	受信ステータス	ラインステータスリード	受信バッファオーバーラン パリティエラー フレーミングエラー ブレーク信号検出
100		データ受信	受信データリード	データ受信
010	↓	送信ホールディング・レジスタ	送信データライト 割り込み参照レジスタリード	送信データ書き込み可
000	低	モデム・ステータス	モデム・ステータスリード	CD/CI/DSR/CTS

1-3　ラインコントロールレジスタ

1-4　モデムコントロールレジスタ

＊1：＊は負論理を表す
＊2：各ビットとも1で対応する出力端子がアクティブとなる
＊3：割り込み使用時にはOUT2をセットする

1-5　ラインステータスレジスタ

＊1：ビット0は受信データリードによりリセット
＊2：ビット1～4はラインステータスレジスタリードによりリセット
＊3：ビット5は送信データライトによりリセット
＊4：送信バッファはダブルバッファであり，ビット5はバッファが1つ以上空きで送信データを書き込み可能なことを示し，ビット6はダブル・バッファがともに空きであることを示す

1-6　モデムステータスレジスタ

＊：ビット3－0は過渡状態のCD/CI/DSR/CTS信号

139〜233行：X68000依存コード
235〜311行：PC-9801依存コード
327〜401行：IBM PC依存コード

詳しくは説明しませんが，X68000用のブロック送信の一部172〜185行までを見てください。

相手方は，送ったデータを正しく受信できたかどうか応答してきます。そのコードを，ここで受信処理をしています。正常受信を示すコード，異常受信を示すコードのいずれかが返されるのです。ただし，通信異常でまったくデータが流されないことも考え，5秒以内で応答コードがないときにはタイムアウトとみなし，異常終了するようになっています。

対して，IBM PC用の391〜397行を見てください。ずいぶんと簡素化されているのがわかると思います。ここでは無条件に1文字受信を行い，正常（0が応答）か異常（1が応答）かの判定を行っています。5秒間のタイムアウトチェックなどは一切やっていません。

当然，きちんとしたプログラミングは，前者です。逆に後者はいいかげんなプログラミングの見本です。しかし，IBM PCで実際に安定して動作するのは後者です。X68000では問題ないのですが，IBM PCにおいて高速マシンで動作してもHP200LXのような低速マシンでは，正常に動作しないのです。つまり，タイムアウト処理をしている間に肝心の通信データを取りこぼしてしまうわけです。

さて，リスト2に今回の従機側プログラムの変更箇所を示します。3月号掲載“R.C”の512〜723行までのブロックを，リスト2と差し替えてください。

つまり，

```
512 ：/****************************************/
513 ：/* dskini  ディスク初期化ルーチン     */
514 ：/****************************************/
                    ⋮
721 ：/****************************************/
722 ：/*GETDPB DPBテーブル取得ルーチン*/
723 ：/****************************************/
```

以上をリスト2に入れ替えます。

リスト2は，従機プログラムの若干の仕様変更と，IBM PC用の制御機能の追加を行っています。ここではi8250の機能説明をかねて機種依存部を紹介します。

まず，330〜340行，関数_rs_inz()で，i8250の初期化を行います。

336行のoutp()関数はC言語より，ポートアドレスへデータの書き込みを行っています。第1パラメータにポートアドレスを，第2パラメータに，書き込みたい内容をセットしています。ここでは，define定数を使っているため，具体的な値はわかりにくいかもしれません。定数の値は，320〜329行で定義していますから参照してください（これ以降の説明は，必ず図1と見比べてみましょう）。

そして，336行では，0x3f9番地に0を書き込んでいます。0x3f9番地は割り込み許可レジスタです。ここに0を書き込むことにより，ポーリングによる通信を指定しています。

同様にして337行でポートアドレス0x3fcに，3 (00000011_B)を書き込むRS-232Cの通信のハードウェアフローを行うRTSとCTSを設定しています。

338行の_rs_spd()は，i8250へ通信速度を設定します。実体部は342〜359行です。

351〜352行でi8250に実際に設定するクロックの分周値を生成し，変数clkとclk1に格納しています。

353行のinp()は，ポートアドレスからデータを読み取る関数です。第1パラメータでポートアドレスを指定し，戻り値が実際のポートの値です。ここでは0x3fb番地ラインコントロールレジスタの値を取得しています。

354行では同じ0x3fbへ，最上位ビットをオンにした値を書き込んでいます。このビットはDLABと呼ばれ，オンにすることにより，i8250は通信速度設定モードに変わります。これにより，0x3f8，0x3f9に速度に対応する分周値を書き込めば通信速度を設定できます。355〜356行でこの設定を行っています。

357行で再びDLABビットをオフにし，通常モードにしています。

ここまでの説明で，i8250の使い方がだいたいわかると思います。360〜372行の関数_LOF232C()ではi8250に受信データの有無をチェックしています。373〜391行_INP232C()では，受信データの取得。393〜412行では_OUT232C ()で一文字送信を行っています。腕に自信のある方は図1を頼りに追跡してみてください。基本的な考え方はX68000のZ8530と同一であることに気がつくことでしょう。

リスト3に主機側の修正例を示します。なぜか，原稿執筆時に従機側のX68000が起動しなくなりました。いろいろと調べていくと，ケーブルを介して従機がスタンバイしているときは問題なく起動し，そうでないときは起動しないことがわかりました。筆者の記憶では，これまで起動しないときは，リセットしなおせば通常は起動したものでした。気持ちが悪いので，測定機などを介してちょっと調べてみました。まず，仮想ドライブがHuman68kに組み込まれるとき，仮想ドライブはRS-232Cにデータを送り従機から応答の有無をチェックするようになっています。従機がつながっていないときには，ハードフロー信号のCTSがオフになっていることがわかりました。結局，仮想ドライブは，CTSがオンになるまで送信待ちとなり，止まったままになってしまうことになったようです。

これを解決したのが，リスト3です。これは主機のRS-232C初期化プログラムですが，従来きちんと見ていたハードウェアフローを無効にし使わないようにしています。579行からをリストどおりに変更してください。これにより，無条件にRS-232C出力するようになります。

IBM PCでの使い方

使い方は従来どおりで変更はありません。コマンドラインより，

R －DA［CR］

以上のように実行してください。そして，主機側のX68000を再起動してください。“－DA”は仮想化したいドライ

ブを指定しています。ここでは，従機のAドライブ以降をすべて仮想ドライブとして登録することを示します。

ここで注意しなくてはいけないことは，IBM PCは，常にハードディスクがCドライブより始まるということです。X68000やPC-9801では，起動ドライブがAドライブになるという特性があります。ですからFDで起動すれば，FDがAドライブ，そして，ハードディスクがCドライブ以降に続きます。対してIBMでは，FDであろうとハードディスクであろうと常にドライブ番号は固定化されています。

また，IBM系では，FDが1機しかないときには，AドライブとBドライブが同じFDドライブに二重定義されるということです。1ドライブのIBM PCで次のように実行する動作がわかりやすいでしょう。

C>COPY A:＊.＊ B:[CR]

ここで，AとBは同じドライブですがエラーにならず実行できます。その代わり，転送元ファイルをAドライブから読み取り，Bドライブに書き込むとき，画面上に「BドライブにFDをセットしてください」というメッセージが出てきます。つまり，1ドライブをAとBで使えるということは，そのたびにディスク交換することを前提にしています。

ここで，副作用があります。当仮想ドライブシステムで，Aドライブからの利用を指定すると，A,B,C……という順序で登録を開始します。ですから，Aドライブを登録し終わったときに，（Aドライブと共有している）Bドライブの登録を行います。するとDOSが，「ディスクをBドライブにセットしてください」というメッセージを出しキー入力待ちとなり止まります。DOSは，Bドライブとして扱うフロッピーをドライブにセットするのを待っているわけです。このキー入力待ちが，仮想ドライブの登録をストップさせます。

これは，もう対処のしようがないので，IBM PC上でAドライブから登録するときには，ディスク交換のメッセージが出たら，すぐにリターンキーを打ち，処理が止まらないようにしてください。面倒であればBドライブから指定するのが無難でしょう。

当システムの指定したドライブ以降を登録するという仕様は，X68000⟷X68000での利便性を考えたものですが，IBM PCでは見事に裏目に出ています。起動時のオプションをIBM PCでも使いやすくなるよう変更すべきかもしれませんね。

HP200LXでの場合

HP200LXはFDが存在しないので，従機側プログラムを通信でまず流し込む必要があります。HP200LXとX68000をまず，通信ケーブルで接続してください。ケーブルはヒューレットパッカード純正の「100LX/200LX<=>PC/AT互換機」接続ケーブル（型番F1015）を用意しておく必要があります。このケーブルは，X68000とは直接つながらないので，9ピン25ピンメスメス変換アダプタを介してX68000のRS-232Cコネクタに接続します。このようなまわりくどいことをするのは，HP200LXの通信コネクタが特殊であるため，IBM PC接続専用の純正ケーブルしか入手できないからです。モデムケーブルと間違えないよう注意してください。

そして，次にはパソコン通信を利用して従機側プログラムをHP200LXに流し込む必要があります。HP200LXでは標準で添付されている通信ソフトDataComを利用するのがよいでしょう。X68000側はみなさんの使い慣れた通信ソフトで結構です。双方の通信ソフトで，手続きを統一し転送してください。これは，XMDOEMあたりが無難でしょう。

このあとは，DOSのコマンドラインより，従機側プログラムを実行すればよいわけです。ドライブの指定は，Cドライブより指定してください。ちなみにHP200LXのドライブは次のとおりです。

C:内蔵RAMディスク
D:内蔵ROMディスク
E:ICカードディスク

そして，Eドライブは，標準的なIBM PCマシンと互換性をとるため，Aドライブとしても認識するようになっています。ですから，仮想登録にCドライブを指定すれば，Eドライブとして通常のAドライブをアクセスできるようになります。逆に，Aドライブから指定するとBドライブの登録で障害が発生します。

大容量ディスクでの注意

今回の開発にあたり，コンパックについている200Mバイトハードディスクを仮想ドライブで利用できないという障害が発見されました。なぜか，このハードディスクをアクセスしようとするとディスクアクセスエラーが出ます。いろいろと調べた結果，Human68k（ここでは，ver.2）のブロック型デバイスの管理情報であるbpbテーブルに問題があることがわかりました。bpb上では，ディ

HP200LXについて

HP200LXを知らない方のことも考えて，もう少しこのマシンを詳しく説明しましょう。

このマシンはインテルi80186というCPUを搭載しています（i8086と同性能のものと思ってください）。80186は，8086の周辺LSIを1つのパッケージに統合してまとめたCPUです。ですから，処理能力そのものは同等です。

クロックは8MHzです。IBM PCは，初代IBM PC(i8088)，IBM PC/XT(8086)，IBM PC/AT (80286)……というシリーズ展開をしています。ですからちょうど，IBM PC/XT相当の製品と思ってください（カッコ内は採用しているCPU）。

また，NECのPC-9801でいうならば，初代9801 (i8088)，PC-9801E, F, M (i8086)……という流れできていますから，PC-9801E, F, M相当のマシンといえるでしょう。

かのマシンたちのデビューより10年の歳月が流れ，同じものがここまで小さくなったわけです。

メモリは2Mバイト搭載していますが，古典的生粋DOSマシンなので，640Kバイトしかアプリケーションからは使用できません。残りのメモリはRAMディスクとして使用します。また，増設メモリ用にPCM-CIA仕様のICカードスロットが1ついています。通常は，ここに5Mバイトから20Mバイトのメモリカードを差し，ハードディスクの代わりに使用します。

このマシンをストレスなくX68000の携帯端末としてコキ使うためには，日本語化キットを別途組み込む必要があります。このあたりの説明は本編の役割ではないので割愛します。興味のある方は各自調べてみてください。

スクの最大セクタ数（ここでいうセクタは実際のセクタではなくクラスタのことと思われる）を2バイトのメモリで管理しています。それが，かのディスクでは，セクタ数が多すぎて2バイトではあふれてしまうのが原因でした。

これにより，対策はもうお手上げ状態です。OSの管理手法によりいたしかたないのでしょう。コンパックではDOS ver.5が載っています。MS-DOSが，ver.3,4,5,6と上がるにつれて拡張され，このようなことが起きたのだと思います。

ただ，このままでは都合が悪いので，仮想ドライブを起動するときに，ドライブの最大セクタ数を調べ，X68000で扱えるかチェックする機能をつけました。これにより最大セクタ数が大きすぎる場合は，エラーとみなし，そのディスクは仮想ドライブとして使えなくするように改良しました。

また，運用による対応方法は，ディスクを複数のパーティション分割して，1ドライブあたりのセクタ数を減らせばよいのです。また，1クラスタあたりのサイズを増やせば，結果的にbpbテーブルに収まりますので使えるようになります。

PC-9801系での改良

詳しくは述べませんが，今回のプログラムではPC-9801用の部分も相当改良してあります。いままでのPC-9801用は，BIOSコールを利用して9,600bps固定でした。これを今回，PC-9801が搭載している通信用LSI（i8251）を直接制御するように改良しました。これにより，最高38,400bpsを出せるようになっています。理論上はX68000と76,800bpsまで通信可能ですが，現時点では，76,800bpsでの実験はしていません。

また，以前よりたびたび述べてきたようにPC-9801系は，そのCPUクロックにより，8MHz系と5MHz系の2種類があります。もし，8MHz系であれば，どんなに高速なPC-9801でも通信速度は上限が9,600bpsに制限されます。この場合は今回のプログラムでも高性能化は望めません。ご注意ください。

今回のプログラムは，起動時，始めに9,600bpsでX68000と通信を開始し，従機が8MHz系か5MHz系かをチェックします。そして，もしも5MHz系であれば通信速度を38,400bpsへアップしてX68000と交信するように改造されています。

また，今回説明しなかったi8251の使い方は，i8250と大同小異と思ってください。

PC-9801系での改良動機

当初，PC-9801を改良する予定ではありませんでした。すでに，低速とはいえ，BIOSコール版が動いていたからです。それより，まだ，開通していないIBM PCとの接続が重要であると考えていました。

しかし，実際にIBM PCとの接続を開始するといくつかの問題が出てきました。最初の試作品では9,600bps，19,200bpsでは動きましたが，38,400bpsでは動作しませんでした。とりあえず，これはプログラムにCPUパワーをムダにする要素があり，通信でデータのとりこぼしが発生しているせいと判断しました。

そして，プログラムのぜい肉を削り高速化しました。それでもなかなか動作しません。もしやと思いそれまでテストで利用していたコンパックの80486DX2（50MHzクロック）マシンをやめて，HP200LXでテストしてみました。結果は上々で38,400bpsでなんら問題なく動作します。LXとコンパックでは20倍くらい処理能力に開きがあります。やはり，なんらかのタイミングが問題なのでしょう。それも，CPUが速すぎるのが原因と思われます。それならばということで，LXと同クラスの処理能力をもつ，エプソンPC-286LH10（V30，10MHz）ではどうだろうと，機種こそ違え同じDOSマシンで，通信LSIを直叩きするよう急遽プログラムを書き換えてみました。

Z-Link 限定誌上配布

約1年にわたり作ってきた仮想ドライブシステムもようやく，X68000←→X68000，X68000←→PC-9801，X68000←→IBM PCとのファイル互換ができるようになりました。つきましては，これまでの成果をFDにて有料配布します。

配布FDには，次のものが入ります。
- 全ソースコード
- X68000用実行形式プログラム
- PC-9801互換機用実行形式プログラム
- IBM PC互換機用実行形式プログラム

<申込方法>

郵便振替により，セット内容を指定のうえ送金してください。各セットは送料，消費税込の価格です。各セットの違いは，添付ケーブルの相違です。住所・氏名・電話番号，セット番号を必ず明記してください。発送は送金日より2週間くらいかかります。ご了承ください。

●Aセット内容
（X68000←→X68000，X68000←→PC-9801接続用）
- 5″2HDディスク×1
- 3.5″2HDディスク×1
- 1.5mクロスケーブル（25ピンオスオス）×1

注）ケーブルは，X68000←→X68000，X68000←→PC-9801接続用です（PC-9801note除く）

料金：8,000円

●Bセット内容
（X68000←→IBM PC互換接続用）
- 5″2HDディスク×1
- 3.5″2HDディスク×1
- 1.5mクロスケーブル（25ピンオスオス）×1

注）ケーブルは，X68000←→X68000，X68000←→PC-9801接続用です（PC-9801note除く）
- IBM PC用通信コネクタ変換アダプタ（XT用）×1
- IBM PC用通信コネクタ変換アダプタ（AT用）×1

料金：11,000円

郵便振替口座番号
00130-6-536826

有効期限
1995年7月31日まで

注意）今回は，FDのみの配布はいたしません。これは，通信ケーブルに雑多な種類があるため，動作保証できないためです。セットに添付したケーブルのみ動作保証をします。また，HP100/200LXの方はHP純正のケーブルを別途購入してご利用ください。AないしBセット添付のケーブルは使えません。

〈お問い合わせ〉
有限会社電机本舗
東京都港区高輪1-2-16鈴木ビル6A

そしたら，問題なく38,400bpsで動作します。これより，なんらかの理由でコンパックが（CPUが速すぎて）カラ振りしていると推定できます。

詳しく調べてみたところ，やはりコンパックが速く動き過ぎてタイミングが合わないのが原因でした。コンパックがデータのブロック送信を完了した時点で，X68000より応答コードを待ちます。このときに，応答コードが戻ってこない内に，さっさとタイムアウトを起こしてしまったのです。そこで，相手の受信部の遅延要素（厳密には，通信エラーが起きたとき，通信データを空にする機能）を削除して動くようにしました。

高速の世界ではいろいろと奇妙なことが起きます。特に，標準的な通信制御用の測定機は，19,200bpsまでしか対応していません。これ以下であれば，2台のパソコンの間に測定機を入れ，データの流れを見れば，どこで狂い出しているのか手に取るようにわかります。しかし，測定機の使えない世界では手探りですからあの手この手のデバッグとなります。このようなわけで，IBM PCの動作検証のために高速版を作らざるを得なくなったため，PC-9801版も高速化が行われたのです。

〈参考文献〉

「MS-DOS完全活用法」中島信行著，CQ出版社

リスト1

```
  1: #define         X68     0
  2: #define         PC98    1
  3: #define         IBM     2
  4: #define         SYSTEM  1
  5:
  6: #if SYSTEM==X68
  7: #include        <doslib.h>
  8: #include        <stdio.h>
  9: #include        <time.h>
 10: #endif
 11:
 12: #if SYSTEM==PC98 || SYSTEM==IBM
 13: #include        <stdlib.h>
 14: #include        <dos.h>
 15: #include        <sys¥types.h>
 16: #include        <sys¥stat.h>
 17: #define         NULL    0
 18: #endif
 19:
 20: #define TIMEOUT 5L
 21: #define BLK_LEN 1024
 22:
 23: int     blk_in();
 24: int     blk_in0();
 25: int     blk_in1();
 26: int     blk_out();
 27: int     blk_out0();
 28: int     blk_out1();
 29: void    _rs_buf_clr();
 30:
 31: int     _LOF232C();
 32: int     _INP232C();
 33: void    _OUT232C();
 34: /*************************************************************
 35:         blk_in  複数データを受信
 36: *************************************************************/
 37: int     blk_in( data, len )
 38: unsigned char   *data;                  /* 転送データ格納アドレス */
 39: int     len;                            /* 転送データ長 */
 40: {
 41:   int   sts;
 42:   int   i;
 43:   int   n;
 44:   unsigned char *p;
 45:         n = BLK_LEN;
 46:         p = data;
 47:         for( i=0 ; i<len ; i+=BLK_LEN ) {
 48:                 if( (i+BLK_LEN)>len ) {
 49:                         n = len - i;
 50:                 }
 51:                 sts = blk_in0( p, n );
 52:                 if( sts ) {
 53:                         break;
 54:                 }
 55:                 p += BLK_LEN;
 56:         }
 57:         return( sts );
 58: }
 59: /*************************************************************
 60:         blk_out 複数データを送信
 61: *************************************************************/
 62: int     blk_out( data, len )
 63: unsigned char   *data;                  /* 転送データ格納アドレス */
 64: int     len;                            /* 転送データ長 */
 65: {
 66:   int   sts;
 67:   int   i;
 68:   int   n;
 69:   unsigned char *p;
 70:         n = BLK_LEN;
 71:         p = data;
 72:         for( i=0 ; i<len ; i+=BLK_LEN ) {
 73:                 if( (i+BLK_LEN)>len ) {
 74:                         n = len - i;
 75:                 }
 76:                 sts = blk_out0( p, n );
 77:                 if( sts ) {
 78:                         break;
 79:                 }
 80:                 p += BLK_LEN;
 81:         }
 82:         return( sts );
 83: }
 84: /*************************************************************
 85:         blk_in0 複数データを受信
 86:         受信に失敗したら5回までやりなおす
 87: *************************************************************/
 88: int     blk_in0( data, len )
 89: unsigned char   *data;                  /* 転送データ格納アドレス */
 90: int     len;                            /* 転送データ長 */
 91: {
 92:   int   sts;
 93:   int   i;
 94:         for( i=0 ; i<5 ; i++ ) {
 95:                 sts = blk_in1( data, len );
 96:                 if( sts==0 ) {
 97:                         break;
 98:                 }
 99:                 _rs_buf_clr();  /* 通信バッファクリア */
100:         }
101:         return( sts );
102: }
103: /*************************************************************
104:         blk_out0        複数データを送信
105:         送信に失敗したら5回までやりなおす
106: *************************************************************/
107: int     blk_out0( data, len )
108: unsigned char   *data;                  /* 転送データ格納アドレス */
109: int     len;                            /* 転送データ長 */
110: {
111:   int   sts;
112:   int   i;
113:   int   c;
114:         for( i=0 ; i<5 ; i++ ) {
115:                 _rs_buf_clr();  /* 通信バッファクリア */
116:                 sts = blk_out1( data, len );
117:                 if( sts==0 ) {
118:                         break;
119:                 }
120:         }
121:         return( sts );
122: }
123: /*************************************************************
124:         _rs_buf_clr     通信バッファクリア
125: *************************************************************/
126: void _rs_buf_clr()
127: {
128:   int   c;
129:   int   i;
130:         xenable();
131:         for( i=0 ; i<2000 ; i++ ) {
132:                 while( _LOF232C() ) {
133:                         c = _INP232C();
134:                 }
135:         }
136:         xdisable();
137: }
138: #if SYSTEM==X68
139: /*************************************************************
140:         blk_out1        複数データ送信
141: *************************************************************/
142: int     blk_out1( data, len )
143: unsigned char   *data;                  /* 転送データ格納アドレス */
144: int     len;                            /* 転送データ長 */
145: {
146:   time_t        tm;
147:   time_t        tm1;
148:   time_t        tmx;
149:   int   bsc;
150:   int   i;
151:   unsigned char *ptr;
152:   int   c;
153:   int   sts;
154:   int   n;
155:         xenable();
156:         sts = -1;
157:         bsc = 0;
158:         ptr = (unsigned char*)data;
159:         for( i=0 ; i<len ; i++ ) {
160:             if( _LOF232C() ) {
161:                                         /* 同調機構,受信側でとりこぼし*/
162:                 return(sts); /* 強制リターン */
163:             }
164:             c = *ptr;
165:             _OUT232C( c );
166:             bsc ^= c;
167:             ptr ++;
168:         }
169:         xdisable();
170:         _OUT232C( bsc );
171:         tm = time( NULL );
172:         while(1) {
173:                 tm1 = time( NULL ); /* TimeOut チェック */
174:                 tmx = tm1 - tm;
```

```
175:                 if( tmx> TIMEOUT ) {
176:                         break;
177:                 }
178:                 if( _LOF232C() ) {
179:                         c = _INP232C();
180:                         if( c==0 ) {
181:                                 sts = 0;
182:                         }
183:                         break;
184:                 }
185:         }
186:         return( sts );
187: }
188: /****************************************************************
189:         blk_inl 複数データ受信
190: ****************************************************************/
191: int     blk_inl( data, len )
192: unsigned char   *data;                  /* 転送データ格納アドレス */
193: int     len;                            /* 転送データ長 */
194: {
195:     time_t          tm;
196:     time_t          tml;
197:     time_t          tmx;
198:     int     bsc;
199:     int     i;
200:     unsigned char *ptr;
201:     int     c;
202:     int     sts;
203:     int     n;
204:         sts = -1;
205:         bsc = 0;
206:         ptr = (unsigned char *)data;
207:         n = 0;
208:         for( i=0 ; i<4000000L ; i++ ) {
209:                 B_BPOKE(0xe98005, 0);
210:                 if( B_BPEEK(0xe98005)&0x01 ) {
211:                         c = B_BPEEK(0xe98007);
212:                         *ptr++ = c;
213:                         bsc ^= c;
214:                         i = 0;
215:                         n++;
216:                         if( n==len ) {
217:                                 break;
218:                         }
219:                 }
220:         }
221:         c = _INP232C();
222:         xenable();
223:         if( c!=bsc ) {
224:                 _OUT232C( 1 );
225:         }
226:         else {
227:                 _OUT232C( 0 );
228:                 sts = 0;
229:         }
230:         xdisable();
231:         return( sts );
232: }
233: #endif
234: #if SYSTEM==PC98
235: /****************************************************************
236:         blk_outl                複数データ送信
237: ****************************************************************/
238: int     blk_outl( data, len )
239: unsigned char   *data;                  /* 転送データ格納アドレス */
240: int     len;                            /* 転送データ長 */
241: {
242:     time_t          tm;
243:     time_t          tml;
244:     time_t          tmx;
245:     int     bsc;
246:     int     i;
247:     unsigned char *ptr;
248:     int     c;
249:     int     sts;
250:     int     n;
251:         xenable();
252:         sts = -1;
253:         bsc = 0;
254:         ptr = (unsigned char*)data;
255:         for( i=0 ; i<len ; i++ ) {
256:                 if( _LOF232C() ) {
257:                                 /* 同調機構,受信側でとりこぼし*/
258:                         return(sts); /* 強制リターン */
259:                 }
260:                 c = *ptr;
261:                 _OUT232C( c );
262:                 bsc ^= c;
263:                 ptr ++;
264:         }
265:         xdisable();
266:         _OUT232C( bsc );
267:         c = _INP232C();
268:         if( c==0 ) {
269:                 sts = 0;
270:         }
271:         return( sts );
272: }
273: /****************************************************************
274:         blk_inl 複数データ受信
275: ****************************************************************/
276: int     blk_inl( data, len )
277: unsigned char   *data;                  /* 転送データ格納アドレス */
278: int     len;                            /* 転送データ長 */
279: {
280:     int     bsc;
281:     int     c;
282:     long    i;
283:     int     sts;
284:     char    *max;
285:         sts = -1;
286:         max = data;
287:         max += len;
288:         bsc = 0;
289:         for( i=0 ; i<1000000L ; i++ ) {
290:                 if( inp(0x32)&0x02 ) {
291:                         *data = (unsigned char)inp( 0x30 );
292:                         bsc ^= *data;
293:                         data ++;
294:                         if( data==max ) {
295:                                 break;
296:                         }
297:                         i = 0L;
298:                 }
299:         }
300:         c = _INP232C();
301:         xenable();
302:         if( c!=bsc ) {
303:                 _OUT232C( 1 );
304:         }
305:         else {
306:                 _OUT232C( 0 );
307:                 sts = 0;
308:         }
309:         xdisable();
310:         return( sts );
311: }
312: #endif
313: #if SYSTEM==IBM
314: #define COM_PORT        0x3F8 /* COM1=3F8,COM2=2F8,COM3=3E8,CO
M4=2E8*/
315: #define IOP_IER         1       /* interrupt enable register*/
316: #define LCR_DLAB_BIT    0x80    /* divisor addres latch bit*/
317: #define IOP_TX_DATA     0
318: #define IOP_RX_DATA     0
319: #define LCR_BITS        3       /* 8 data, no parity, no break*/
320: #define IOP_LCR         3       /* line control register*/
321: #define IOP_LSR         5       /* line status register*/
322: #define LSR_RX_READY    1
323: #define LSR_TX_READY    32
324: /**************************************************************
***
325:         blk_outl                複数データ送信
326: **************************************************************
**/
327: int     blk_outl( data, len )
328: unsigned char   *data;                  /* 転送データ格納アドレス */
329: int     len;                            /* 転送データ長 */
330: {
331:     time_t          tm;
332:     time_t          tml;
333:     time_t          tmx;
334:     int     bsc;
335:     int     i;
336:     unsigned char *ptr;
337:     int     c;
338:     int     sts;
339:     int     n;
340:         xenable();
341:         sts = -1;
342:         bsc = 0;
343:         ptr = (unsigned char*)data;
344:         for( i=0 ; i<len ; i++ ) {
345:                 if( _LOF232C() ) {
346:                         /* 同調機構、受信側でとりこぼし */
347:                         return(sts); /* 強制リターン */
348:                 }
349:                 c = *ptr;
350:                 _OUT232C( c );
351:                 bsc ^= c;
352:                 ptr ++;
353:         }
354:         xdisable();
355:         _OUT232C( bsc );
356:         c = _INP232C();
357:         if( c==0 ) {
358:                 sts = 0;
359:         }
360:         return( sts );
361: }
362: /****************************************************************
363:         blk_inl 複数データ受信
364: ****************************************************************/
365: int     blk_inl( data, len )
366: unsigned char   *data;                  /* 転送データ格納アドレス */
367: int     len;                            /* 転送データ長 */
368: {
369:     int     bsc;
370:     int     c;
371:     long    i;
372:     int     sts;
373:     char    *max;
374:       sts = -1;
375:       max = data;
376:       max += len;
377:       bsc = 0;
378:       for( i=0 ; i<1000000L ; i++ ) {
379:           if( inp(COM_PORT+IOP_LSR)&LSR_RX_READY ) {
380:             *data = (unsigned char)inp( COM_PORT+IOP_RX_DATA );
381:             bsc ^= *data;
382:             data ++;
383:             if( data==max ) {
384:               break;
385:             }
386:             i = 0L;
387:           }
388:       }
389:       c = _INP232C();
390:       xenable();
391:       if( c!=bsc ) {
392:           _OUT232C( 1 );
393:       }
394:       else {
395:           _OUT232C( 0 );
396:           sts = 0;
397:       }
398:       xdisable();
399:       return( sts );
400: }
401: #endif
```

▶仕事でMacintoshとDOS/V機を使っていますが，やはりX68000がいちばん使いやすいと思います。といいつつも，最近はSEGA SATURNで遊んでばかりいます……。

牧野　豊(24)茨城県

リスト2

```
  1: /*******************************************
  2:        dskini  ディスク初期化ルーチン
  3: *******************************************/
  4: int r_dskini( req )
  5: struct REQ_INI *req;
  6: {
  7:    int    sts;
  8:    struct        DPBPTR  d;
  9: #if SYSTEM==PC98 || SYSTEM==IBM
 10:    struct        DPBPTR1 *d1;
 11: #endif
 12:    struct        BPB_TBL bpb_tbl;
 13:    int    fat;
 14:    int    fat_no;
 15:    int    rsv_sct;
 16:    char   dsk_flg;
 17:    int    drv;
 18:    char   d_no;
 19:    long   mode;
 20:    long   old_drv;
 21:    long   wk_drv;
 22:    int    ver;
 23:    int    sub_ver;
 24:    UBYTE  id;
 25:    long   max_wk;
 26:    long   max_wk1;
 27: #if SYSTEM==X68
 28:    _OUT232C( BPS76800 );
 29:    dlytime();
 30:    _rs_spd( BPS76800 );
 31:    printf( "Speed up to 76,800bps¥n¥r" );
 32:    dlytime();
 33: #endif
 34: #if SYSTEM==PC98
 35:    _OUT232C( _get_spd() );
 36:    dlytime();
 37:    _rs_inz( _get_spd() );
 38:    if( _get_spd()==BPS38400 ) {
 39:       printf( "Speed up to 38,400bps¥n¥r" );
 40:    }
 41: #endif
 42: #if SYSTEM==IBM
 43:    _OUT232C( BPS38400 );
 44:    dlytime();
 45:    _rs_inz( BPS38400 );
 46:    printf( "Speed up to 38,400bps¥n¥r" );
 47:    dlytime();
 48: #endif
 49:    drv = _drv;
 50:    old_drv = CURDRV();         /* 現在のドライブ保存   */
 51:    while( 1 ) {
 52:       CHGDRV( (long)drv-1 );
 53:       wk_drv = CURDRV();
 54:       wk_drv++;
 55:       if( drv != wk_drv ) {
 56:          dsk_flg = -1;
 57:          sts = blk_out( &dsk_flg, sizeof(dsk_flg) );
 58:          break;
 59:       }
 60:       mode = 0L;
 61:       mode = DRVCTRL( mode, drv );
 62:       mode &= 2L;
 63:       d_no = 'A' + (char)drv;
 64:       d_no --;
 65:       sts = GETDPB( drv, &d );
 66:       id = d.id;
 67: #if SYSTEM==PC98 || SYSTEM==IBM
 68:       getver( &ver, &sub_ver );
 69:       if( ver>4 ) {
 70:          d1 = (struct DPBPTR1*)(&d);
 71:          id = d1->id;
 72:       }
 73: #endif
 74:       if( mode==0 ) {
 75:          bpb_tbl.b_no = 1024; /* セクタあたりのバイト数 */
 76:          bpb_tbl.sct_no = 1;  /* クラスタあたりのセクタ数 */
 77:          bpb_tbl.fat_no = 2;  /* ファット領域の個数 */
 78:          bpb_tbl.rsv_sct_no = 1; /* 予約領域のセクタ数 */
 79:          bpb_tbl.root_ent_no = 192; /* ルートの最大ファイル数
*/
 80:          bpb_tbl.sct_max = 1232; /* 全セクタ数 */
 81:          bpb_tbl.id = 0xfe;      /* メディアバイト */
 82:          bpb_tbl.fat_sct_no = 2; /* 1fatのセクタ数 */
 83:       }
 84:       else if( id==0xfe || id==1 ) {   /* $$ */
 85:          bpb_tbl.b_no = 1024;/* セクタあたりのバイト数 */
 86:          bpb_tbl.sct_no = 1;  /* クラスタあたりのセクタ数 */
 87:          bpb_tbl.fat_no = 2;  /* ファット領域の個数 */
 88:          bpb_tbl.rsv_sct_no = 1;  /* 予約領域のセクタ数 */
 89:          bpb_tbl.root_ent_no = 192; /* ルートの最大ファイル数
*/
 90:          bpb_tbl.sct_max = 1232; /* 全セクタ数 */
 91:          bpb_tbl.id = 0xfe;      /* メディアバイト */
 92:          bpb_tbl.fat_sct_no = 2; /* 1fatのセクタ数 */
 93:       }
 94:       else {
 95: #if SYSTEM==X68
 96:          /* セクタあたりのバイト数 */
 97:          bpb_tbl.b_no = d.byte;
 98:          /* クラスタあたりのセクタ数 */
 99:          bpb_tbl.sct_no = d.sec + 1;
100:          /* ファット領域の個数 */
101:          bpb_tbl.fat_no = d.fatcount;
102:          /* 予約領域のセクタ数 */
103:          bpb_tbl.rsv_sct_no = d.fatsec;
104:          /* ルートの最大ファイル数 */
105:          bpb_tbl.root_ent_no = d.dircount;
106:          max_wk = (unsigend long)(d.maxfat);
107:          max_wk1 = (unsigend long)(d.sec+1);
108:          max_wk *= max_wk1;
109:          if( max_wk>65535L ) {
110:             bpb_tbl.sct_max = 0; /* 全セクタ数size over! */
111:          }
112:          else {
113:             /* 全セクタ数set */
114:             bpb_tbl.sct_max = (UWORD)(max_wk);
115:          }
116:          /* 全セクタ数 */
117:          bpb_tbl.sct_max = d.maxfat*(d.sec+1);
118:          bpb_tbl.id = d.id;        /* メディアバイト */
119:          bpb_tbl.fat_sct_no = d.fatlen; /* 1fatのセクタ数 */
120: #endif
121: #if SYSTEM==PC98 || SYSTEM==IBM
122:          getver( &ver, &sub_ver );
123:          if( ver<4 ) {
124:             /* セクタあたりのバイト数 */
125:             bpb_tbl.b_no = d.byte;
126:             /* クラスタあたりのセクタ数 */
127:             bpb_tbl.sct_no = d.sec + 1;
128:             /* ファット領域の個数 */
129:             bpb_tbl.fat_no = d.fatcount;
130:             /* 予約領域のセクタ数 */
131:             bpb_tbl.rsv_sct_no = d.fatsec;
132:             /* ルートの最大ファイル数 */
133:             bpb_tbl.root_ent_no = d.dircount;
134:             max_wk = (unsigned long)(d.maxfat);
135:             max_wk1 = (unsigned long)(d.sec+1);
136:             max_wk *= max_wk1;
137:             if( max_wk>65535L ) {
138:                /* 全セクタ数size over! */
139:                bpb_tbl.sct_max = 0;
140:             }
141:             else {
142:                /* 全セクタ数set */
143:                bpb_tbl.sct_max = (UWORD)(max_wk);
144:             }
145:             /* メディアバイト*/
146:             bpb_tbl.id = d.id;
147:             /* 1fatのセクタ数 */
148:             bpb_tbl.fat_sct_no = d.fatlen;
149:          }
150:          else {
151:             d1 = (struct DPBPTR1*)(&d);
152:             /* セクタあたりのバイト数 */
153:             bpb_tbl.b_no = d1->byte;
154:             /* クラスタあたりのセクタ数 */
155:             bpb_tbl.sct_no = d1->sec + 1;
156:             /* ファット領域の個数 */
157:             bpb_tbl.fat_no = d1->fatcount;
158:             /* 予約領域のセクタ数    */
159:             bpb_tbl.rsv_sct_no = d1->fatsec;
160:             /* ルートの最大ファイル数         */
161:             bpb_tbl.root_ent_no = d1->dircount;
162:             max_wk = (unsigned long)(d1->maxfat);
163:             max_wk1 = (unsigned long)(d1->sec+1);
164:             max_wk *= max_wk1;
165:             if( max_wk>65535L ) {
166:                bpb_tbl.sct_max = 0;/* 全セクタ数size over! */
167:             }
168:             else {
169:                /* 全セクタ数set */
170:                bpb_tbl.sct_max = (UWORD)(max_wk);
171:             }
172:             /* メディアバイト */
173:             bpb_tbl.id = d1->id;
174:             /* 1fatのセクタ数 */
175:             bpb_tbl.fat_sct_no = d1->fatlen;
176:          }
177: #endif
178:       }
179:       dsk_flg = 0;
180:       if( bpb_tbl.sct_max ) {
181:          /* bpb tbl 送信    */
182:          if( (sts=blk_out( &dsk_flg, sizeof(dsk_flg) )) ) {
183:             break;
184:          }
185:          else {
186:             /* もしDOSなら配列変換        */
187:             XCHG2( &(bpb_tbl.b_no) );
188:             XCHG2( &(bpb_tbl.rsv_sct_no) );
189:             XCHG2( &(bpb_tbl.root_ent_no) );
190:             XCHG2( &(bpb_tbl.sct_max) );
191:             /* bpb tbl 送信     */
192:             if( (sts=blk_out(&bpb_tbl, sizeof(bpb_tbl) )) ) {
193:                break;
194:             }
195:             /* $$もしDOSなら配列変換         */
196:             XCHG2( &(bpb_tbl.b_no) );
197:             _byte[drv-_drv] = bpb_tbl.b_no;
198:             drv ++;
199:             printf( "%c: を主機へ仮想ドライブとして登録 ¥n", d
_no );
200:          }
201:       }
202:       else {
203:          printf( "%c: はセクタ数が多すぎて仮想ドライブに登録で
きません ¥n", d_no );
204:          drv ++;
205:       }
206:    }
207:    CHGDRV( old_drv );        /* ドライブ復旧 */
208:    return( sts );
209: }
210: /*******************************************
211:     PC9801機種依存関数
212: *******************************************/
213: #if SYSTEM==PC98
214: int   _max_spd;
215: /*******************************************
216:    _get_spd     最高速度取得
217:    return   :   speed
218: *******************************************/
219: int _get_spd()
220: {
221:    struct  SREGS   s_reg;
222:    unsigned char p;
```

▶CD-ROMを買うべきか迷っています。またCD-ROMの特集をやってください。
伊藤　則広(24)愛知県

```
223:    segread( &s_reg );
224:    movedata( 0,0x501, s_reg.ds, (unsigned int)(&p), 1 );
225:    _max_spd = BPS38400;
226:    if( p & 0x80 ) {
227:       _max_spd = BPS9600;
228:    }
229:    return( _max_spd );
230: }
231: int   _spd98_10[] = { 128, 64, 32, 16, 8, 4, 2, 1, 0 };
232: int   _spd98_8[] = { 104, 52, 26, 13 };
233: char      __dmy[128];
234: /****************************************
235:    _rs_inz       初期化
236:    return   :   sts
237: ****************************************/
238: int      _rs_inz( spd )
239: int   spd;
240: {
241:   union   REGS   i_reg;
242:   union   REGS    o_reg;
243:   struct  SREGS   s_reg;
244:   int   sts;
245:    segread( &s_reg );
246:    i_reg.h.ah = 0;
247:    i_reg.h.al = 0x07;
248:    i_reg.h.ch = 0x4e;
249:    i_reg.h.cl = 0x37;
250:    s_reg.es = s_reg.ds;
251:    i_reg.x.di = (int)__dmy;
252:    i_reg.x.dx = 128;
253:    i_reg.h.bh = 0x2;
254:    i_reg.h.bl = 0x1e;
255:    int86x( 0x19, &i_reg, &o_reg, &s_reg );
256:    outp( 0x77, 0xb6 );
257:    dlytime();
258:    if( _max_spd == BPS9600 ) {
259:       outp( 0x75, _spd98_8[spd] );
260:    }
261:    else {
262:       outp( 0x75, _spd98_10[spd] );
263:    }
264:    dlytime();
265:    outp( 0x75, 0x00 );
266:    dlytime();
267:    outp( 0x32, 0x27 ); /*****/
268:    outp( 0x37, 0x92 );
269:    dlytime();
270:    outp( 0x37, 0x07 );
271:    dlytime();
272:    return( i_reg.h.ah );
273: }
274: /**********************************************
275:    _LOF232C   受信バッファチェックルーチン
276: **********************************************/
277: int   _LOF232C()
278: {
279:   int   sts;
280:    sts = 0;
281:    if( inp(0x32)&0x02 ) {
282:       sts = 1;
283:    }
284:    return( sts );
285: }
286: /****************************************
287:    _INP232C   1文字受信ルーチン
288: ****************************************/
289: int   _INP232C()
290: {
291:   long   i;
292:   int   sts;
293:    sts = -1;
294:    for( i=0 ; i<20000L ; i++ ) {
295:       if( inp(0x32)&0x02 ) {
296:          sts = (unsigned char)inp( 0x30 );
297:          break;
298:       }
299:    }
300:    return( sts );
301: }
302: /****************************************
303:    _OUT232C   1文字送信ルーチン
304: ****************************************/
305: void   _OUT232C( int c )
306: {
307:   long   i;
308:   int   sts;
309:    sts = -1;
310:    for( i=0 ; i<1000000L ; i++ ) {
311:       if( inp(0x32)&0x01 ) {
312:          outp( 0x30, c );
313:          sts = 0;
314:          break;
315:       }
316:    }
317: }
318: #endif
319: #if SYSTEM==IBM
320: #define   COM_PORT      0x3F8 /* COM1=3F8,COM2=2F8,COM3=3E8,COM
4=2E8 */
321: #define   IOP_IER       1     /* interrupt enable register */
322: #define   LCR_DLAB_BIT 0x80  /* divisor addres latch bit */
323: #define   IOP_TX_DATA  0
324: #define   IOP_RX_DATA  0
325: #define   LCR_BITS      3  /* 8 data, no parity, no break */
326: #define   IOP_LCR       3  /* line control register */
327: #define   IOP_LSR       5  /* line status register */
328: #define   LSR_RX_READY 1
329: #define   LSR_TX_READY 32
330: /****************************************
331:    _rs_inz       初期化
332:    return   :   sts
333: ****************************************/
334: int _rs_inz( int spd )
335: {
336:    outp( COM_PORT+IOP_IER, 0 );
337:    outp( COM_PORT+IOP_IER+IOP_LCR, LCR_BITS );
338:    _rs_spd( spd );
339:    return( 0 );
340: }
341: int _clk_tbl[] = { 96, 48, 24, 12, 6, 3, 2, 1 };
342: /****************************************
343:    _rs_spd       rs232c speed change
344:    return   :   sts
345: ****************************************/
346: int _rs_spd( unsigned int bps )
347: {
348:   unsigned int    clk;
349:   unsigned char   clk1;
350:   unsigned char   b;
351:    clk = _clk_tbl[bps];   /* クロックset      */
352:    clk1 = (unsigned char)( clk >> 8 );
353:    b = inp( IOP_LCR + COM_PORT );
354:    outp( IOP_LCR + COM_PORT, (unsigned char)(b | LCR_DLAB_BIT)
);
355:    outp( COM_PORT, (unsigned char)clk );
356:    outp( COM_PORT + 1, clk1 );
357:    outp( COM_PORT + IOP_LCR, (unsigned char)b );
358:    return(0);
359: }
360: /****************************************
361:    _LOF232C   受信buffer check
362:    return    : char, or sts
363: ****************************************/
364: int _LOF232C()
365: {
366:   int   sts;
367:    sts = 0;
368:    if( inp(COM_PORT+IOP_LSR)&LSR_RX_READY ) {
369:       sts = 1;
370:    }
371:    return( sts );
372: }
373: /********************************************
374:    _INP232C       ＲＳ２３２Ｃ一文字受信
375:    return    : char, or sts
376: ********************************************/
377: int   _INP232C()
378: {
379:   long   i;
380:   int   sts;
381:   long  ii;
382:    sts = -1;
383:    ii = 2000;
384:    for( i=0 ; i<ii ; i++ ) {
385:       if( inp(COM_PORT+IOP_LSR)&LSR_RX_READY ) {
386:          sts = (unsigned char)inp( COM_PORT+IOP_RX_DATA );
387:          break;
388:       }
389:    }
390:    return( sts );
391: }
392: /**/
393: /****************************************
394:    _OUT232C   ＲＳ２３２Ｃへ 一文字送信
395:    in        : c    送信文字
396:    return    : no
397: ****************************************/
398: void _OUT232C( int c )
399: {
400:   long   i;
401:   int   sts;
402:    sts = -1;
403: /*   while( (inp(COM_PORT+IOP_LSR) & LSR_TX_READY)==0 );
404: */
405:    for( i=0 ; i<1000000L ; i++ ) {
406:       if( inp(COM_PORT+IOP_LSR) & LSR_TX_READY ) {
407:          outp( COM_PORT+IOP_TX_DATA, (unsigned char)c );
408:          sts = 0;
409:          break;
410:       }
411:    }
412: }
413: #endif
414: #if SYSTEM==PC98 || SYSTEM==IBM
415: /****************************************
416:    GETDPB   ＤＰＢテーブル取得ルーチン
417: ****************************************/
```

リスト3

```
579: /*****************************************************
580:    _rs_inz          初期化
581:    return  :        sts
582: *****************************************************/
583: _rs_inz( spd )
584: int         spd;              /* speed */
585: {
586:       __mode = SET232C( -1 );
587:       SET232C( 0x4c07 );
588:       B_BPOKE(0xe98005, 0x03); /* 受信モードセット */
589:       B_BPOKE(0xe98005, 0xc1); /* 0xe1->0xc1フローをみないよう変更 *
/
590:       B_BPOKE(0xe98005, 0x05); /* 送信モードセット */
```

〈対応機種一覧〉●MZ-80 K/C/700/1500 ●MZ-80 B/2000 ●MZ-2500/2861 ●X 1 ●X 1 turbo/Z ●PC-8001/8801/88 ●SMC-777/C ●PASOPIA/5 ●PASOPIA/7 ●FM-7/77/AV ●MSX/2/2+/turbo R ●PC-286/386/486/9801/98/9821 ●X 68000/X 68030
掲載されたプログラムの利用には各機種用のS-OS“SWORD”システムが必要です。

第158部　FE ver.1.0

●スクリーンエディタFE ver.1.0

今月は，タブコード対応のスクリーンエディタFE ver.1.0を掲載します。

ESC(SHIFT＋BREAK)シーケンスを用いたごく基本的なもので，特殊文字(タブ，改行，エンドコード）の表示機能をもつスクリーンエディタです。

RUN&SUBMITの拡張がしてあれば，ファイルを指定したエディタの起動ができたり，文字単位のカット&ペーストができたりと基本は押さえています。問題があるとすれば,ESCキーがない機種(コントロールコードを入力するたびに，SHIFT＋BREAKを押すのは面倒くさいかもしれない),そしてファイル入力周りでしょうか。

そして，人によっては，コントロールコードの充実も必要かもしれません。個人的にEMATEをぎたぎたにいじくり回して遊んだことがあるので，ちょっとだけ指がムズムズしてしまいます。読者の皆さんもプログラムでなくてもいいですから，なにかアイデアが浮かんだならば，このTHE SENTINELまでお寄せください。

また，このエディタでは，きちんとタブコードが管理されています。そのため，REDAのエディタやEDC-Tで問題となった文字列中のスペースがタブコードに変換されてしまうということはありません。安心してご使用ください。

また,いままでS-OS用として発表されたスクリーンエディタと同様に，実行速度はそれなりに遅いです。そこで，FEでも各機種専用のラインルーチンを用意することで，高速化できるようになっています。

今月はX1/turbo専用ルーチンのみですが，これ以外でも投稿がありしだいどんどん掲載していく予定です。

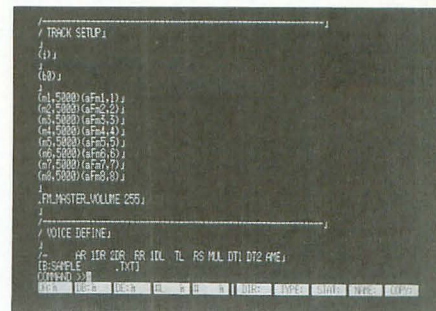
なお，ラインプリントルーチンの詳細は，来月号で行いたいと思います。もちろん，解析の得意な人であれば，さほど苦労せずに同等のものを制作することができると思います。

大きいプログラムは苦手でも，ちょっとしたサブルーチン程度のものなら作れるぞ，という人はがんばって挑戦してみてください。

●PICTパズル(仮名)

6月号で提案した「お絵描きパズル」ですが,「運を天に任せる会社が怖くてできなかった」というハガキをいただきました。確かにゲームボーイ用に発売されてはいるのですが，元のパズルは任○堂が考えたわけではありません(したがって任○堂にパズルの著作権があるとは思えない)。

商品名(製品名)をそのまま使うのはまずいのですが，パズルのルールをそのまま使ってもなんの問題もないはずです。実際，各種パズル専門誌はもちろんのこと，一般誌，マンガ雑誌にまで掲載されています。

実は，このハガキをくださったのは「BLOCK DOWN」制作者の春名さん。なんでも，「PICTパズル」の制作をやろうかな，と思い立ったはいいけど，先ほど紹介した理由で制作を取り止めてしまったとか。う～ん，これはもったいない。確かに任○堂ということで，恐れる気持ちはわかりますが，上記のようにパズルを制作するだけなら特に問題はありません。安心して制作に打ち込んでください。

もちろん，そのほかにも制作を考えている方もがんばってくださいね(ちなみに8月号の付録ディスクではX68000版「PICTパズル」が収録される予定)。なお「お絵描きパズル」という名称は雑誌名で使われているので，今月からOh!Xでは「PICTパズル」という名称を使います。

ということで，いまが旬のこのパズル，せっかくですからS-OSでも遊びたいじゃないですか。THE SENTINELでは，読者の挑戦を待っています。

●今月のフリーソフト

今月フリーソフトとして連絡があったのが，6月号で紹介した「BLOCK DOWN」です。制作者の春名さん，ご協力感謝します。

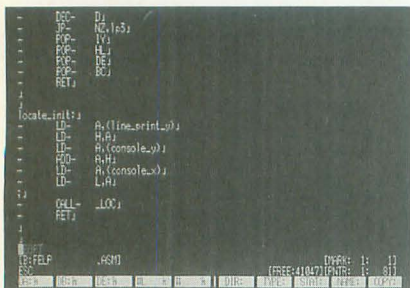
THE SENTINELでは，引き続きフリーソフト化に協力していただける方を募集します。いままで制作したS-OS用アプリケーションをコピーしてもいいよ，という人がいらっしゃいましたら，アンケートハガキでご連絡ください。こちらのほうもよろしくお願いします。

1995■インデックス

全機種共通
S-OS"SWORD"要

FE ver.1.0

Matsufuji Hideshi
松藤 秀史

ESCシーケンスによるコントロールコード，タブコード対応のフルスクリーンエディタが登場です。このFEでは，いままでのエディタにあったタブコード展開の問題点も解決しています。

今回制作したプログラムは，「Full Screen Editor for S-OS "SWORD" FE ver. 1.00」(以下FE)という，S-OS"SWORD"環境下で動くスクリーンエディタです。

ESCシーケンスを用いた，シングルウィンドウのエディタで，あまり，ごてごてとした機能をつけずに，なるべくすっきりとした簡単なものを目指して制作しました。

入力方法

このプログラムは，エディタ本体とラインプリントルーチンからなっています。まず，リスト1のメイン部分をMACINTO-Cなどのツールを使って打ち込んでください。

チェックサムを確認したあと，

#S FE.OBJ:3000:47FF:3000

として，いったんデバイスにセーブしましょう。次にラインプリントルーチン（リスト2が全機種共通システム用，リスト3がX1/turbo専用です）も同じようにMACINTO-Cなどを使って打ち込み，セーブしてください(X1/turbo専用ルーチンのときはエンドアドレスを4470にすること)。

#S FELP.OBJ:4400:444B

そして，

#L FE.OBJ
#L FELP.OBJ
#S FE.OBJ:3000:47FF:3000

以上のようにメインプログラムとラインプリントルーチンをメモリにロードして，一緒にセーブすればできあがりです。

使用方法

FEではスクリーンエディットモードとコマンドモードに分かれています。FEを起動すると，まずスクリーンエディットモードになります。

そのときにRUN&SUBMITの拡張を行っていれば，起動時に編集したいファイル名を指定することもできます。

例） # FE.OBJ FE.ASM

機能説明

現在FEがサポートしている機能を説明していきます。

●エディットモード

実際にテキストの編集を行うモードです。

●コマンドモード

コマンドモードでは，ファイルの操作，テキストの検索，置換などを行います。スクリーンエディットモードからESC+8でコマンドモードに移り，画面のいちばん下にコマンドプロンプトが表示されます。なお，使えるコマンドは表1のとおりです。

●ESC(SHIFT+BREAK)シーケンス

FEでは，ほかのエディタと同様にESCシーケンスを採用しています。使えるコントロールコードは表2のとおりです。

そして，このESCキーには2つの役割があります。ひとつは，エディタにコントロールコードを送ることです(ESCモード)。ESCモードでは，カーソル移動に関するコントロールコードが入力され続けるかぎり，モードを継続するようになっています。そのほかのコントロールコードが入力された場合は，実行後ESCモードを抜けます。また，ESCモード中にもう一度ESCキーを押すとESCモードを抜けます。

そして，もうひとつの機能は，ブロックオペレーションモードです。詳しい内容は，次のカット，コピー，ペースト機能を読んでください。

●カット，コピー，ペースト機能

カット，コピー，ペースト機能は文字単位でサポートしています。これらの機能を利用するには，まず，ESCキーを押します。すると，カーソル位置を示すカラムの上にマーク位置が表示されます（この状態をブロックオペレーションモードといいます)。この状態でカーソルを移動し，カット，コピーする領域を指定します。

要するに，ESCキーを押したときのカーソル位置が，カットもしくはコピー機能の先頭位置の指定を兼ねている，ということを覚えておいてください。

なお，テキストを操作するコントロールコードを入力すると，ブロックオペレーションモードを抜けます。

●ファインド，リプレイス機能

使用法は従来のエディタとほとんど同じですが，タブコードも検索，置換文字に使うことができます。また，セパレータは，ダブルクォーテーション以外の文字であれば，なんでもかまいません。あとファインドモアという機能は，いちばん最後に入力された検索文字列を参照して検索を行うものです。

●特殊文字の表示

FEでは特殊文字(改行コード，タブコード，EOF）を表示する機能がついています。起動時にはすべての特殊文字が表示されていますが，改行コード，タブコードについては，ESC+4，ESC+5で非表示にすることができます。また，コンフィグレー

図　メモリマップ

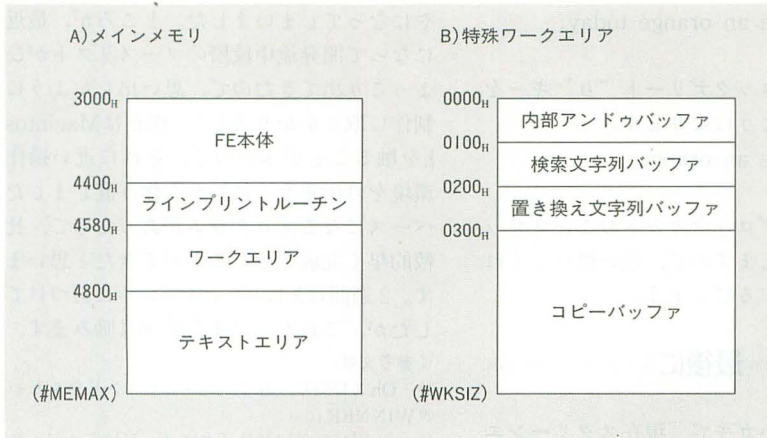



表1　コマンド一覧

コマンド	機能
E[n]	コマンドモードを抜け，エディットモードに移る。nを指定するとn行へカーソルを移動してからエディットモードに移る。E0でテキストの最後へカーソルを移動する
Q	ファイルを閉じてエディタを終了する
L file	現在編集しているテキストを閉じて，fileをロードする
S [file]	現在編集しているテキストをファイル名fileでセーブする
N [file]	現在編集しているテキストを閉じて新規ファイルfileを編集する
F "str"	現在位置から文字列strを検索する
FM	現在位置から以前入力された検索文字列strを検索する
C "str1" "str2"	現在位置から文字列str1をstr2に置換する
B	現在位置をテキスト先頭とする

表2　コントロールコード一覧

キー	機能番号	機能
→ ESC+D	1	カーソルを右へ移動する
← ESC+S	2	カーソルを左へ移動する
↑ ESC+E	3	カーソルを上へ移動する
↓ ESC+X	4	カーソルを下へ移動する
ESC+W	5	1ページ上へ移動する
ESC+Z	6	1ページ下へ移動する
ESC+F	11	カーソルを1ワード右へ移動する
ESC+A	12	カーソルを1ワード左へ移動する
ESC+,	24	カーソルを行の最後へ移動する
ESC+.	23	カーソルを行の先頭へ移動する
ESC+M	15	改行
ESC+I	13	タブ
ESC+O ESC+R	9	インサート/デリートモードの切り替え
ESC+H	14	バックスペース
ESC+0	16	ブロックデリート
ESC+1	17	ブロックカット
ESC+2	18	ブロックコピー
ESC+3	19	ブロックペースト
ESC+4	20	改行コード表示/非表示
ESC+5	10	タブコード表示/非表示
ESC+6	22	タブ幅4/8文字切り替え
ESC+8	21	コマンドモード
ESC+9	8	エディタ終了

ションにより，特殊文字の置き換え文字を変更することができます。

●そのほか

タブ幅は4文字または8文字に固定されています（ESC+6で切り替え）。また，1行255文字以内に制限されています。255文字を超えて文字を入力しようとすると警告を発しますが，置き換えなどによって255文字を超えてしまった場合は，自動的に改行コードを挿入するようになっています。

ラインプリントルーチン

このFEは，もともと機種別のラインプリントルーチンを作成することを前提に作られています。そのため，標準のラインプリントルーチンのままでは非常に低速です。改造する部分は，次の2カ所です。

1)　プログラムの1文字表示部分(CALL_PRINT)

2)　表示位置初期化部分(locate_init)

詳しい内容はリスト5，6を見てのとおりです。なお，多色表示が可能な機種については，本体プログラムワークエリア内のchr_atriにキャラクターのカラーコードを格納しているので，特殊文字の色づけが可能となります。

コンフィグレーション

FEでは，ワークエリアを直接書き換えることにより，各種設定を変更することができます。変更できるものは以下のとおりです。

1)　テキストの格納アドレス

2)　起動時のタブ幅

3)　カーソルが画面横からはみ出したときの画面の移動量

4)　タブコードの置き換え文字およびカラーコード

5)　改行コードの置き換え文字およびカラーコード

6)　空白文字

7)　そのほかの特殊文字の置き換え文字およびカラーコード

8)　EOFの置き換え文字およびカラーコード

9)　新規ファイルのファイル名

10)　コントロールコードの割りつけ

なおタブコードの置き換え文字と改行コードの置き換え文字のカラーコードは同一になります。それぞれのアドレス，詳しい内容は表3にまとめておきましたので参照してください。

応用

FEは基本的な機能しか備えていません。したがって，行削除やカーソル以降の文字列削除，カーソル位置文字削除は，コントロールコードの組み合わせで実現します。

・行削除

表3　コンフィグレーション

アドレス	初期値	機能
3003H	4800H	テキストの格納アドレス
3005H	8	起動時のタブ幅
3006H	20	横方向の画面の移動量
3007H	"－"	タブコードの置き換え文字
3008H	"」"	改行コードの置き換え文字
3009H	" "	空白文字
300AH	"."	そのほかのコードの置き換え文字
300BH	"[EOF]", 26, 0, 0	ファイル終端コード(00H)の置き換え文字
3013H	3	そのほかのコードのカラーコード
3014H	5	EOFのカラーコード
3015H	6	タブ，改行コードのカラーコード
3016H	7	通常文字のカラーコード
3017H	"UNTITLED.", 0	新規ファイルのファイル名
3029H ⋮ 3068H		コントロールコードのキー割りつけ

注）　EOFの置き換え文字の最後には必ず26Hを入れてください。キー割りつけの並びは@，A～Z，という順番になっています。該当するキーのアドレスに機能番号を書き込むようにしてください。

行削除をするためには，まずカーソルを行の先頭に移動します。

I don't have an orange today.
[EOF]

ここでESCキーを押して，カーソルを下へ移動します。

I don't have an orange today.
[EOF]

そして，ブロックデリート“0”キーを押すと行削除ができます。

・カーソル以降の文字列削除

削除したい位置にカーソルを移動し，ESCキーを押します。

I don't have an orange_today.
[EOF]

ここで“.”を押し，行の最後へ移動します。

I don't have an orange today_
[EOF]

そして，ブロックデリート“0”キーを押すと以下のようになります。

I don't have an orange_
[EOF]

同様にしてブロックカットおよびブロックコピーも行えますので，使い慣れてくればかなり重宝するでしょう。

最後に

アンケートハガキで「現在スクリーンエディタを制作中」という経過報告をしてから，3年ほどたちました。あのときは，ソースリストを紛失してしまったのでうやむやになってしまいました。ところが，最近になって開発途中段階のソースリストがひょっこり出てきたので，思い出したように制作に取りかかりました。現在はMacintoshを触ることが多いので，それに近い操作環境を目指そうと最初から作り直しました。ベースになるプログラムがあったので，比較的早く完成させることができたと思います。2週間はX1のディスプレイに釘づけでしたが，これからはまた勉学に励みます。

<参考文献>
1) Oh!X1988年8月号「マルチウィンドウエディタWINNER」
2) Oh!X1990年11月号「タブコード対応エディタEDC-T」
3) MIA X1マシン語プログラミング入門

リスト1 FE.OBJ

```
3000 C3 E3 30 00 48 08 14 B0 : EA
3008 A3 20 A5 5B 45 4F 46 5D : FA
3010 1A 00 00 03 05 06 07 55 : 84
3018 4E 54 49 54 4C 45 44 20 : 34
3020 20 20 20 20 2E 20 20 20 : 0E
3028 00 00 0C 00 00 01 03 0B : 1B
3030 00 0E 0D 00 00 00 0F 00 : 2A
3038 09 00 00 09 02 00 00 00 : 14
3040 05 04 00 06 07 01 02 03 : 1C
3048 04 00 00 00 00 00 00 00 : 04
3050 00 00 00 00 00 18 00 17 : 2F
3058 00 10 11 12 13 14 0A 16 : 7A
3060 00 15 08 00 00 00 00 00 : 1D
3068 00 7B B7 C2 C2 30 7E 23 : 87
3070 FD 23 FE 20 D0 FE 09 CA : DF
3078 92 30 FE 0D CA 9E 30 B7 : 1C
---------------------------------
SUM: 8F 7C 23 E2 84 BC 9A 81 A0BB

3080 CA AA 30 FE 1A CA B6 30 : 6C
3088 3A 13 30 32 D4 45 3A 0A : 0C
3090 30 C9 1E 01 3A 15 30 32 : C9
3098 D4 45 3A 07 30 C9 1E 02 : 73
30A0 3A 15 30 32 D4 45 3A 08 : 0C
30A8 30 C9 21 0B 30 3A 14 30 : D3
30B0 32 D4 45 C3 6E 30 1E 02 : CC
30B8 3A 16 30 32 D4 45 3A 09 : 0E
30C0 30 C9 FE 01 CA CB 30 3A : F7
30C8 09 30 C9 FD 7E 00 B7 C2 : F6
30D0 D8 30 FD 23 3A 09 30 C9 : 64
30D8 1E 00 3A 16 30 32 D4 45 : E9
30E0 C3 6E 30 11 FB 41 CD E5 : 60
30E8 1F DD 21 E4 45 CD 4A 31 : 8E
30F0 CD 65 31 CD C4 31 CD E7 : D9
30F8 31 CD 03 32 3A 05 30 3D : DF
---------------------------------
SUM: ED 39 01 95 8E 2B E3 F5 5C19

3100 4F CD EF 31 CD DF 38 ED : 0D
3108 5B 76 1F 1A FE 23 28 17 : 6A
3110 1A 13 B7 28 12 FE 20 20 : 5C
3118 F7 CD 5E 40 38 18 CD 8E : 0D
3120 40 DA 1D 41 C3 3D 31 11 : BA
3128 17 30 CD 52 40 11 2A 42 : 23
3130 CD E5 1F C3 3D 31 FE 08 : 08
3138 C2 33 20 18 F0 CD 28 32 : 44
3140 3E 0C CD F4 1F C9 37 C9 : F3
3148 B7 C9 2A 6A 1F 22 82 45 : 1C
3150 22 88 45 ED 4B 03 30 ED : 47
3158 43 80 45 ED 43 84 45 ED : EE
3160 42 22 8E 45 C9 AF 32 91 : 72
3168 45 32 92 45 32 9B 45 32 : 92
3170 9D 45 3A 5C 1F 32 93 45 : A1
3178 D6 07 32 A3 45 32 A7 45 : 15
---------------------------------
SUM: F5 C2 59 E2 70 84 AD 74 493D

3180 D6 04 32 A1 45 32 A5 45 : 0E
3188 D6 06 32 97 45 32 99 45 : FA
3190 D6 06 32 9F 45 D6 06 32 : 00
3198 95 45 3A 5B 1F 3D 32 9E : 9B
31A0 45 32 96 45 32 98 45 32 : 93
31A8 A2 45 32 A4 45 32 A0 45 : 19
31B0 32 A0 45 3D 32 94 45 32 : 91
31B8 9C 45 32 9A 45 32 A6 45 : 0F
31C0 32 A8 45 C9 AF 32 A9 45 : B7
31C8 32 AE 45 32 B3 45 21 00 : 70
31D0 00 22 AA 45 22 AF 45 2A : 51
31D8 84 45 22 AC 45 CD 96 39 : 78
31E0 CD BD 3C CD 9F 3A C9 AF : E4
31E8 32 D6 45 32 D5 45 C9 21 : 83
31F0 FC 45 06 00 78 A1 20 04 : 84
31F8 3E 01 18 01 AF 77 23 04 : A5
---------------------------------
SUM: ED 47 04 DE 40 91 C0 C8 B09E

3200 20 F2 C9 CD FA 3D 2A 68 : 71
3208 1F 11 00 03 B7 ED 52 22 : 4B
3210 DF 45 21 00 00 22 D7 45 : 83
3218 11 00 01 19 22 D9 45 19 : 84
3220 22 DB 45 19 22 DD 45 C9 : 68
3228 AF 32 CA 45 CD 96 3E CD : 5E
3230 9F 3E CD 3E 3B CD 4E 39 : 77
3238 DD 36 01 00 DC 9F 3E CD : 9A
3240 8C 3F CD 2C 3B CD A6 3A : AC
3248 CD DC 3A CD AD 3A 18 06 : B5
3250 CD AD 3A CD 9F 3A 3A CA : 5E
3258 45 B7 CC 7E 3F CD 0F 3B : 9C
3260 DC B6 32 CD DF 3E CD C4 : 3F
3268 32 FE 20 DA 94 32 4F 3A : 79
3270 D6 45 B7 79 20 13 3A D5 : 8D
3278 45 B7 79 20 06 CD 14 3D : B9
---------------------------------
SUM: 10 F8 57 09 38 62 18 D9 EFF4

3280 C3 50 32 CD 4C 3D C3 50 : AE
3288 32 FE 80 D2 CE 32 FE 40 : C0
3290 38 02 E6 1F 21 29 30 16 : CF
3298 00 5F 19 7E F5 3A CA 45 : 34
32A0 B7 28 05 21 B8 43 18 03 : 1B
32A8 21 76 43 F1 CB 27 5F 19 : 35
32B0 5E 23 56 62 6B E9 CD 13 : 6D
32B8 39 DD 36 01 01 CD 9F 3E : F8
32C0 CD 0F 3B C9 2A B5 45 CD : D1
32C8 1E 20 CD 21 20 C9 C3 50 : 28
32D0 32 CD 83 3C CD 91 3B DA : 31
32D8 50 32 CD F4 40 DA 50 32 : DF
32E0 C9 CD 91 3B DA 50 32 CD : 8B
32E8 5A 3A DA 43 33 C3 32 32 : 0B
32F0 CD 91 3B DA 50 32 CD 47 : 09
32F8 3A DA 64 33 C3 32 32 CD : 9F
---------------------------------
SUM: 33 ED E7 56 96 52 94 94 B4DC

3300 52 38 C3 50 32 CD 5A 38 : 2E
3308 C3 50 32 CD 91 3B DA 50 : 08
3310 32 3A 94 45 47 CD 5A 3A : ED
3318 05 20 FA C3 32 32 CD 91 : A4
3320 3B DA 50 32 3A 94 45 47 : F1
3328 CD 47 3A 05 20 FA C3 32 : 62
3330 32 CD BE 39 D2 50 32 CD : 17
3338 91 3B DA 50 32 CD 47 3A : 76
3340 DA 50 32 CD 52 38 CD 9F : 1F
3348 3A C3 32 32 CD BE 39 C3 : E8
3350 50 32 CD D1 39 D2 50 32 : AD
3358 CD 91 3B DA 50 32 CD 5A : 1C
3360 3A DA 32 32 CD 3E 3B CD : 8B
3368 5A 38 CD AD 3A CD 9F 3A : EC
3370 C3 35 32 CD D1 39 C3 50 : 14
3378 32 CD DD 39 D2 50 32 C3 : 2C
---------------------------------
SUM: D1 F5 1F 74 EC 40 CE DB A6BD

3380 37 33 CD 0D 3A D2 50 32 : D2
3388 C3 58 33 CD 91 3B DA 50 : 11
3390 32 3A D6 45 B7 28 06 CD : 39
3398 83 3C C3 32 32 3D 32 D6 : 2B
33A0 45 CD 96 3C CD 4E 3F C3 : 01
33A8 32 32 3A D6 45 B7 28 06 : 9E
33B0 CD 83 3C C3 50 32 3D 32 : 40
33B8 D6 45 CD 4E 3F C3 50 32 : BA
33C0 3A D5 45 EE 01 32 D5 45 : 8F
33C8 CD 83 3C CD 4E 3F C3 50 : F9
33D0 32 CD 83 3C 3A 07 30 47 : 76
33D8 3A 5A 43 32 07 30 78 32 : EA
33E0 5A 43 DD 36 01 01 CD 9F : 1E
33E8 3E C3 35 32 CD 83 3C 3A : 2E
33F0 08 30 47 3A 5B 43 32 08 : 91
33F8 30 78 32 5B 43 DD 36 01 : 8C
---------------------------------
SUM: 0C F5 44 9A 51 B8 07 42 F1E7

3400 01 CD 9F 3E C3 35 32 CD : A2
3408 8E 3C 0E 09 C3 76 32 CD : 19
3410 8E 3C 3A AE 45 B7 CA 59 : D1
3418 34 CD 78 3C 54 5D 1B 7E : FF
3420 12 FE 0D CA 2E 34 B7 CA : CA
3428 2E 34 23 13 18 F1 CD D1 : 3F
3430 39 CD 08 3D C3 50 32 3A : CA
3438 AE 45 B7 CA 50 32 CD 78 : 3B
3440 3C 54 5D 1B 7E 12 FE 0D : A3
3448 CA 53 34 B7 CA 53 34 23 : 7C
3450 13 18 F1 CD D1 39 C3 50 : 06
3458 32 CD 8E 3C CD 91 3B DA : 3C
3460 50 32 CD 5A 3A DA 32 32 : 21
3468 CD 3E 3B AF 3D 32 B3 45 : 5C
3470 CD DC 3A CD AD 3A CD 9F : 03
3478 3A 2A B1 45 CD 71 3A 11 : E3
---------------------------------
SUM: E7 58 51 0B 4F 4C E8 3F DCF3

3480 01 00 CD 0D 3C CD 05 39 : 22
3488 C3 2F 32 CD 8E 3C CD 91 : 19
3490 3B DA 50 32 CD D4 38 E5 : 55
3498 11 01 00 CD DA 3B E1 DA : AF
34A0 50 32 36 0D CD 3E 3B DD : E8
34A8 36 01 00 CD 9F 3E C3 31 : D5
34B0 33 3A D5 45 B7 28 D4 CD : 07
34B8 5A 38 C3 31 33 CD 91 3B : 52
34C0 DA 50 32 CD BD 3C CD 6C : 5B
34C8 35 DA 50 32 19 CD 0D 3C : C0
34D0 CD EF 3C CD 9F 3A DD 36 : B1
34D8 01 00 CD 9F 3E CD 83 3C : 37
34E0 C3 32 32 CD 91 3B DA 50 : EA
34E8 32 CD BD 3C CD 6C 35 DA : 40
34F0 50 32 CD B5 3D DA 50 32 : 9D
34F8 19 CD 0D 3C CD EF 3C CD : F4
---------------------------------
SUM: 5E C6 71 8E E2 09 23 E2 7E35

3500 9F 3A DD 36 01 00 CD 9F : 59
3508 3E CD 83 3C CD F5 3D C3 : 8C
3510 32 32 CD 91 3B DA 50 32 : 59
3518 CD BD 3C CD 6C 35 DA 50 : 5E
3520 32 CD B5 3D DA 50 32 DD : 2A
3528 36 01 00 CD 9F 3E CD 83 : 31
3530 3C CD F5 3D C3 32 32 CD : 2F
3538 91 3B DA 50 32 3A CD 45 : 74
3540 B7 C2 50 32 CD 83 3C CD : 54
3548 D4 38 E5 CD D2 3D D1 DA : 78
3550 50 32 2A CB 45 19 CD 60 : 02
3558 3D CD 3E 3B CD AD 3A CD : 04
3560 9F 3A DD 36 01 00 CD 9F : 59
3568 3E C3 35 32 CD 00 3E 2A : 9D
3570 C8 45 3A C4 45 16 00 5F : C5
3578 19 44 4D 2A C2 45 3A BE : D3
---------------------------------
SUM: E7 4B 23 C2 69 DF 8B 10 783A

3580 45 16 00 5F 19 50 59 B7 : 33
3588 ED 52 CA 46 31 54 5D 60 : 91
3590 69 C9 CD 91 3B DA 50 32 : 27
3598 CD 83 3C 3A 59 43 FE 08 : 68
35A0 20 04 3E 04 18 02 3E 08 : C6
35A8 32 59 43 3D 4F CD EF 31 : 47
35B0 DD 36 01 00 CD 9F 3E CD : 8B
35B8 3E 3B C3 50 32 CD 91 3B : 57
35C0 DA 50 32 CD 83 3C CD BD : 72
35C8 3C CD D6 3F AF 3D 32 CA : 06
35D0 45 3A 91 45 C6 0A 32 91 : E8
35D8 45 3A 93 45 D6 0A 32 93 : FC
35E0 45 3A 9E 45 32 B6 45 2A : B9
```

```
35E8 9D 45 CD 1E 20 11 4E 43 : 8F
35F0 CD E5 1F 3E 0D 32 FC 46 : 90
35F8 AF 32 AE 45 C3 50 32 AF : C8
-----------------------------------
SUM: D3 A9 7C 7D 34 D2 24 9F F6AE

3600 32 CA 45 CD E7 3F 3A 91 : FF
3608 45 D6 0A 32 91 45 3A 93 : FA
3610 45 C6 0A 32 93 45 11 FC : 2C
3618 46 CD 4A 36 13 E6 DF FE : 69
3620 45 CA 5E 36 FE 51 CA 93 : 4F
3628 36 FE 4C CA 9A 36 FE 53 : 6B
3630 CA C8 36 FE 4E CA DE 36 : F2
3638 FE 42 CA 7B 36 FE 46 CA : C9
3640 03 37 FE 43 CA 72 37 C3 : B1
3648 CC 35 1A FE 20 C0 FE 09 : 00
3650 C0 13 18 F6 1A FE 20 C8 : E1
3658 FE 09 C8 13 18 F6 CD 4A : 07
3660 36 FE 0D 28 0D CD E0 41 : 64
3668 2B 22 AF 45 CD 96 39 CD : AA
3670 BD 3C CD EF 3C CD 9F 3A : 97
3678 C3 32 32 D5 AF 32 AE 45 : D0
-----------------------------------
SUM: B3 1B 00 5B 1B 86 D8 6F 8D3C

3680 32 B3 45 21 00 00 22 AF : 1C
3688 45 CD 96 39 CD BD 3C D1 : 78
3690 C3 19 36 CD F4 40 DA CC : B9
3698 35 C9 D5 CD F4 40 D1 DA : 7F
36A0 CC 35 CD 4A 36 3E 0C CD : 65
36A8 F4 1F CD 0C 41 DA C0 36 : FD
36B0 CD C4 31 CD 96 3E DD 36 : 76
36B8 01 00 CD 9F 3E C3 CC 35 : 6F
36C0 CD 33 20 CD 48 40 18 EB : 78
36C8 CD 4A 36 1A FE 0D 20 03 : 95
36D0 11 E8 45 3E 0C CD F4 1F : 68
36D8 CD E3 40 C3 B3 36 D5 CD : 3E
36E0 F4 40 D1 CD 4A 36 FE 0D : 5D
36E8 28 0C CD 52 40 CD FB 36 : 9E
36F0 CD C4 31 C3 B3 36 11 17 : 96
36F8 30 18 EF 3E 0C CD F4 1F : 61
-----------------------------------
SUM: 8E EA 17 BE 4E B9 7D E7 0FE9

3700 C3 C0 36 CD 4A 36 FE 0D : 11
3708 CA CC 35 47 E6 DF FE 4D : 22
3710 28 1E 78 13 2A 9D 45 1A : 33
3718 B8 CA CC 35 1A FE 0D CA : 72
3720 CC 35 B8 28 07 CD 9A 1F : 6E
3728 23 13 18 F0 AF CD 9A 1F : 73
3730 CD 63 38 DA CC 35 CD 38 : 48
3738 38 CD D6 3F 2A 9D 45 CD : F3
3740 1E 20 11 8F 42 CD E5 1F : F1
3748 2A B5 45 CD 1E 20 CD 21 : 1D
3750 20 E6 DF FE 45 CA 63 37 : 8C
3758 FE 4D 28 0D FE 43 CA CC : 57
3760 35 18 EB CD E7 3F C3 6F : 5D
3768 36 CD D4 38 23 CD 60 3D : 9C
3770 18 BE AF 32 E1 45 CD 4A : F4
3778 36 FE 0D CA CC 35 47 13 : 66
-----------------------------------
SUM: 80 95 65 F5 7A D8 AA CD D411

3780 2A D9 45 1A B8 CA CC 35 : E5
3788 0E 00 1A FE 0D CA CC 35 : FE
3790 B8 28 08 CD 9A 1F 0C 23 : 9D
3798 13 18 EF AF CD 9A 1F 79 : C8
37A0 32 E2 45 2A DB 45 1A 0D : C4
37A8 00 1A FE 0D CA CC 35 B8 : A8
37B0 28 08 CD 9A 1F 0C 23 13 : F8
37B8 18 EF AF CD 9A 1F 79 32 : E7
37C0 E3 45 CD 63 38 DA 2E 38 : D0
37C8 3A E1 45 B7 20 32 CD 38 : 6E
37D0 38 CD D6 3F 2A 9D 45 CD : F3
37D8 1E 20 11 B0 42 CD E5 1F : 12
37E0 2A B5 45 CD 1E 20 CD 21 : 1D
37E8 20 E6 DF FE 52 CA 00 38 : 37
37F0 FE 53 28 22 FE 43 CA CC : 72
37F8 35 FE 41 CA 20 38 18 C2 : 70
-----------------------------------
SUM: 65 0B 9B F2 DC 64 7B 54 8117

3800 CD 94 38 DA 2E 38 CD D4 : 7A
3808 38 3A E3 45 16 00 5F 19 : 28
3810 CD 60 3D C3 C2 37 CD D4 : C7
3818 38 23 CD 60 3D C3 C2 37 : 81
3820 CD D6 3F CD 1A 3F AF 3D : F4
3828 32 E1 45 C3 00 38 DD 36 : 66
3830 01 00 CD 9F 3E C3 CC 35 : 6F
3838 CD 4E 39 CD 3E 3B CD AD : 14
3840 3A CD 13 39 DD 36 01 00 : 67
3848 CD 9F 3E CD 0F 3B CD 2C : BA
3850 3B C9 AF 32 AE 45 CD AD : 52
3858 3A C9 AF 3D 32 B3 45 CD : E6
3860 DC 3A C9 CD D4 38 ED 5B : 00
3868 D9 45 EB CD 94 1F 23 47 : F3
3870 1A 13 B7 CA 46 31 B8 20 : FD
3878 F7 E5 D5 CD 94 1F B7 28 : 10
-----------------------------------
SUM: 19 CB 9E E4 E7 B7 3F DD 4455

3880 0B 4F 1A 13 23 B9 28 F3 : 7E
3888 D1 E1 18 E4 E1 D1 2B CD : 58
3890 60 3D B7 C9 3A E3 45 47 : C6
3898 3A E2 45 90 28 04 38 14 : 69
38A0 18 21 CD D4 38 ED 5B DB : 35
38A8 45 EB CD 94 1F B7 C8 12 : 41
38B0 23 13 18 F6 ED 44 47 CD : 89
38B8 D4 38 16 00 58 CD DA 3B : 5C
38C0 D8 18 DF 47 CD D4 38 3A : 29
38C8 E2 45 16 00 5F 19 58 CD : DA
```

```
38D0 0D 3C 18 CE 2A B1 45 3A : 89
38D8 AE 45 16 00 5F 19 C9 2A : 74
38E0 84 45 AF 32 90 45 77 CD : C3
38E8 EB 38 C9 2A 84 45 CD 71 : 1D
38F0 3A 30 FB 22 86 45 ED 5B : 9A
38F8 84 45 B7 ED 52 23 22 8A : 8E
-----------------------------------
SUM: 6C 76 43 2E A3 CF 05 9E 4584

3900 45 CD 05 39 C9 2A 88 45 : 10
3908 ED 5B 86 45 B7 ED 52 22 : 2B
3910 8C 45 C9 3A A9 45 67 3A : 63
3918 93 45 3D 6F 3A 06 30 57 : 4B
3920 3A B3 45 5F 94 38 05 95 : F7
3928 30 18 B7 C9 7C 92 38 0B : 19
3930 32 A9 45 67 7B 94 38 F4 : C2
3938 C3 46 31 AF 32 A9 45 C3 : CC
3940 46 31 7C 82 32 A9 45 67 : FC
3948 7B 94 95 30 F5 C9 2A AF : 6B
3950 45 ED 5B AA 45 B7 E5 ED : 05
3958 52 E1 38 2F 06 00 3A 94 : 6E
3960 45 4F 0D E5 EB 09 B7 ED : 1E
3968 52 E1 D2 48 31 B7 ED 42 : 64
3970 ED 5B AA 45 22 AA 45 ED : 35
3978 52 44 4D 2A AC 45 CD 71 : 3C
-----------------------------------
SUM: DE CE 7D 8C 7C 41 6F 73 DF54

3980 3A 0B 78 B1 20 F8 22 AC : 54
3988 45 37 C9 22 AA 45 2A B1 : 31
3990 45 22 AC 45 37 C9 ED 4B : 90
3998 AF 45 CD A5 39 22 B1 45 : B7
39A0 ED 53 AF 45 C9 2A 84 45 : F0
39A8 11 00 00 D5 EB B7 ED 42 : B7
39B0 7C B5 EB D1 C8 CD 71 3A : 2D
39B8 DA 46 31 13 18 ED CD 78 : AE
39C0 3C 7E FE 0D CA 46 31 B7 : BD
39C8 CA 46 31 79 3C 32 AE 45 : 1B
39D0 C9 3A AE 45 B7 CA 46 31 : EE
39D8 3D 32 AE 45 C9 CD 78 3C : AC
39E0 7E FE 0D CA 46 31 7E 23 : 6B
39E8 FE 0D 28 1B B7 28 18 0C : 51
39F0 CD 35 3A 20 F1 7E 23 FE : EC
39F8 0D CA 07 3A B7 CA 07 3A : DA
-----------------------------------
SUM: 29 31 86 0A F9 73 F6 F6 70E9

3A00 0C CD 35 3A 28 EF 0D 79 : E5
3A08 32 AE 45 B7 C9 CD 78 3C : 26
3A10 79 B7 CA 46 31 2B 0D 79 : 22
3A18 B7 28 06 7E CD 35 3A 28 : C7
3A20 F4 2B 0D 7E CD 35 3A 28 : 0E
3A28 06 79 B7 28 03 18 F2 0C : 77
3A30 79 32 AE 45 C9 C5 E5 F5 : 06
3A38 21 5D 43 06 00 3A 5C 43 : A0
3A40 4F F1 ED B1 E1 C1 C9 2A : 73
3A48 B1 45 CD 71 3A D8 22 B1 : 19
3A50 45 2A AF 45 23 22 AF 45 : 9C
3A58 B7 C9 2A AF 45 7C B5 CA : 99
3A60 46 31 2B 22 AF 45 2A B1 : 93
3A68 45 CD 7D 3A D8 22 B1 45 : B9
3A70 C9 7E B7 CA 46 31 23 FE : 60
3A78 0D C8 C3 71 3A ED 5B 84 : 0F
-----------------------------------
SUM: 5F FA B4 53 12 24 E1 24 1B12

3A80 45 E5 B7 ED 52 7C B5 E1 : 32
3A88 CA 46 31 1B 2B E5 B7 ED : 10
3A90 52 7C B5 E1 CA 9D 3A 2B : 30
3A98 7E FE 0D 20 F0 23 C9 3A : BF
3AA0 B3 45 32 B4 45 C9 3A B4 : DA
3AA8 45 32 B3 45 C9 21 FC 46 : 9B
3AB0 3A AE 45 4F 06 00 11 FC : 8F
3AB8 45 79 B7 28 1A 7E FE 09 : 3C
3AC0 28 0D FE 0D 28 11 B7 28 : 58
3AC8 0E 04 13 0D 23 18 EA 13 : 6A
3AD0 04 1A B7 28 FA 18 F4 78 : 7B
3AD8 32 B3 45 C9 21 FC 46 3A : 90
3AE0 B3 45 4F 06 00 11 FC 45 : 9F
3AE8 79 B7 28 1E 7E FE 09 28 : 23
3AF0 0D FE 0D 28 15 B7 28 12 : 46
3AF8 13 0D 04 23 18 EA 0D 13 : 69
-----------------------------------
SUM: 0E 28 20 F3 76 76 C9 B1 7062

3B00 1A B7 20 F6 79 B7 28 02 : 41
3B08 18 F4 78 32 AE 45 C9 3A : AC
3B10 A9 45 6F 3A 91 45 67 3A : 0E
3B18 93 45 3D 57 3A B3 45 95 : 33
3B20 5F D8 92 D2 46 31 7B 84 : 11
3B28 32 B5 45 C9 2A AF 45 ED : 00
3B30 4B AA 45 B7 ED 42 3A 92 : EC
3B38 45 85 32 B6 45 C9 CD 4A : D7
3B40 3B 79 32 CE 45 AF 32 CF : A9
3B48 45 C9 2A B1 45 11 FC 46 : 81
3B50 FD 21 FC 45 06 01 0E 01 : 75
3B58 7E 12 FE 0D C8 B7 C8 FE : E0
3B60 09 28 20 04 28 07 FD 23 : A4
3B68 23 13 0C 18 EB 3E 0D 12 : A2
3B70 11 01 00 E5 CD DA 3B E1 : BA
3B78 D8 36 0D DD 36 01 00 CD : FC
-----------------------------------
SUM: 9F D8 21 70 02 77 AD 4F 2EA4

3B80 9F 3E C9 04 CA 6D 3B FD : 19
3B88 23 FD 7E 00 B7 28 F4 18 : 89
3B90 D7 3A CF 45 B7 C8 CD 2C : 9D
3B98 3C 3A CE 45 90 28 04 38 : 7D
3BA0 15 18 24 2A B1 45 11 FC : 7E
3BA8 46 1A 77 FE 0D 28 26 B7 : E7
3BB0 28 23 23 13 18 F3 ED 44 : BD
```

```
3BB8 16 00 5F 2A B1 45 CD 71 : D3
3BC0 3A CD DA 3B D8 18 DC 16 : FE
3BC8 00 5F 2A B1 45 CD 71 3A : F7
3BD0 CD 0D 3C 18 CE CD 0E 3D : 14
3BD8 B7 C9 E5 2A 8C 45 B7 ED : 04
3BE0 52 E1 38 24 D5 ED 5B 86 : 32
3BE8 45 EB B7 ED 52 44 4D 03 : BA
3BF0 E1 ED 5B 86 45 19 EB ED : E5
3BF8 53 86 45 ED B8 CD 0E 3D : DB
-----------------------------------
SUM: F7 45 B5 A5 EA 38 A4 0E 4104

3C00 CD 05 39 CD 71 3F B7 C9 : 08
3C08 CD C6 3F 37 C9 E5 D5 54 : E0
3C10 5D 2A 86 45 B7 ED 52 44 : 3C
3C18 4D 03 D1 E1 E5 ED 52 54 : 7A
3C20 5D E1 ED B0 1B ED 53 86 : BC
3C28 45 C3 FD 3B 21 FC 46 06 : A9
3C30 00 04 7E 23 FE 0D 28 05 : DD
3C38 B7 28 02 18 F4 78 C9 D5 : 03
3C40 E5 2A 86 45 19 ED 5B 88 : C3
3C48 45 B7 ED 52 E1 D1 D2 46 : 05
3C50 31 B7 C9 21 FC 46 06 01 : 1B
3C58 11 FC 45 7E FE 09 28 0D : 0C
3C60 FE 0D C8 B7 C8 04 CA 46 : 66
3C68 31 13 23 18 EE 04 CA 46 : 81
3C70 31 13 1A B7 28 F7 18 F2 : 3E
3C78 21 FC 46 06 00 3A AE 45 : 96
-----------------------------------
SUM: 8A 8B 05 12 D6 B2 6F BA B6FC

3C80 4F 09 C9 AF 32 D6 45 CD : EA
3C88 8E 3C CD 4E 3F C9 AF 32 : CE
3C90 BD 45 CD 41 3F C9 2A AF : F1
3C98 45 22 C0 45 3A AE 45 32 : CB
3CA0 BE 45 2A B1 45 22 C2 45 : 4C
3CA8 3A B3 45 32 BF 45 AF 3D : 54
3CB0 32 BD 45 CD 34 3F CD 9A : DB
3CB8 3F CD A8 3F C9 2A AF 45 : DA
3CC0 22 C6 45 3A AE 45 32 C4 : 50
3CC8 45 2A B1 45 22 C8 45 3A : CE
3CD0 B3 45 32 C5 45 C9 2A C0 : E7
3CD8 45 22 AF 45 3A BE 45 32 : CA
3CE0 AE 45 2A C2 45 22 B1 45 : 3C
3CE8 3A BF 45 32 B3 45 C9 2A : 5B
3CF0 C6 45 22 AF 45 3A C4 45 : 64
3CF8 32 AE 45 2A C8 45 22 B1 : 2F
-----------------------------------
SUM: 87 7C 2C C8 3F 60 96 96 EDEE

3D00 45 3A C5 45 32 B3 45 C9 : 7C
3D08 AF 3D 32 CF 45 C9 AF 3D : E7
3D10 32 90 45 C9 F5 CD 99 3D : 68
3D18 21 FC 46 24 54 5D 2B 3A : 9D
3D20 AE 45 E5 21 01 01 06 00 : 01
3D28 4F B7 ED 42 44 4D E1 ED : 94
3D30 B8 CD 78 3C F1 77 CD 53 : C1
3D38 3C 38 09 0C 79 32 AE 45 : 27
3D40 CD 08 3D C9 CD A7 3D CD : 59
3D48 B6 3F 37 C9 F5 CD 99 3D : 8D
3D50 CD 78 3C 7E FE 0D CA 18 : EC
3D58 3D B7 CA 18 3D C3 34 3D : 47
3D60 ED 4B AF 45 0B 54 5D 2A : 12
3D68 B1 45 E5 B7 ED 52 F5 7C : 42
3D70 B5 28 21 F1 E1 30 08 CD : D5
3D78 71 3A 03 38 02 18 EB D5 : C0
-----------------------------------
SUM: 89 6C 07 F9 47 CF 33 A9 AE13

3D80 CD 7D 3A D1 ED 43 AF 45 : 79
3D88 22 B1 45 EB B7 ED 52 7D : 76
3D90 32 AE 45 C9 F1 E1 03 18 : DB
3D98 EB 21 FC 46 ED 5B D7 45 : B2
3DA0 01 00 01 CD 97 1F C9 21 : 6F
3DA8 FC 46 ED 5B D7 45 01 00 : A7
3DB0 01 CD 91 1F C9 E5 D5 CD : CE
3DB8 EC 3D 38 10 ED 53 CB 45 : C1
3DC0 42 4B ED 5B DD 45 CD 97 : 5B
3DC8 1F D1 E1 C9 CD C6 3F 37 : A3
3DD0 18 F7 E5 D5 ED 5B CB 45 : 21
3DD8 E5 D5 CD DA 3B C1 E1 38 : 76
3DE0 08 ED 5B DD 45 CD 91 1F : EF
3DE8 B7 D1 E1 C9 E5 2A DF 45 : 65
3DF0 B7 ED 52 E1 C9 AF 32 CD : 4E
3DF8 45 C9 AF 3D 32 CD 45 C9 : 07
-----------------------------------
SUM: 0F A9 34 B9 9D A2 E4 97 8962

3E00 2A C6 45 ED 5B C0 45 B7 : 39
3E08 ED 52 28 0B D8 CD 44 3E : 99
3E10 CD 27 3E CD 61 3E C9 3A : A1
3E18 BE 45 47 3A C4 45 B8 D8 : 1D
3E20 CD 27 3E CD 61 3E C9 3A : A1
3E28 BE 45 47 3A C4 45 32 BE : 7D
3E30 45 78 32 C4 45 3A BF 45 : 36
3E38 47 3A C5 45 32 BF 45 78 : 39
3E40 32 C5 45 C9 2A C0 45 ED : 21
3E48 5B C6 45 ED 53 C0 45 22 : CD
3E50 C6 45 2A C2 45 ED 5B C8 : 4C
3E58 45 ED 53 C2 45 22 C8 45 : BB
3E60 C9 3A AE 45 32 B7 45 3A : 5E
3E68 B3 45 32 BC 45 2A AF 45 : 49
3E70 22 B8 45 2A B1 45 22 BA : 1B
3E78 45 C9 3A B7 45 32 AE 45 : 69
-----------------------------------
SUM: 34 5F D4 2B 68 73 7A 56 9DC5

3E80 3A BC 45 32 B3 45 2A B8 : 47
3E88 45 22 AF 45 2A BA 45 22 : A6
3E90 B1 45 CD 9F 3A C9 3E 0C : AF
3E98 CD F4 1F CD F3 3E C9 2A : D1
```

```
3EA0 AC 45 3A 94 45 4F AF 57 : 59
3EA8 47 32 D1 45 22 D2 45 78 : 40
3EB0 32 D1 45 DD 72 00 DD 7E : F2
3EB8 01 B7 28 12 C5 E5 ED 4B : D4
3EC0 B1 45 B7 ED 42 7C B5 E1 : EE
3EC8 C1 C4 00 44 18 03 CD 00 : B1
3ED0 44 04 CD 71 3A 38 04 0D : 09
3ED8 20 D2 C9 16 02 18 F8 21 : 04
3EE0 FC 46 22 D2 45 3A B6 45 : B0
3EE8 32 D1 45 AF DD 77 00 CD : 18
3EF0 00 44 C9 CD 03 3F CD 1A : 03
3EF8 3F CD 27 3F CD 4E 3F CD : 99
---------------------------------
SUM: 66 1D FC F0 30 19 74 B0 DC4F

3F00 71 3F C9 2A 9B 45 CD 1E : 6E
3F08 20 3E 5B CD F4 1F 11 E8 : 92
3F10 45 CD E5 1F 3E 5D CD F4 : 72
3F18 1F C9 2A 95 45 CD 1E 20 : F7
3F20 11 D7 42 CD E5 1F C9 2A : EE
3F28 97 45 CD 1E 20 11 E4 42 : 1E
3F30 CD E5 1F C9 2A 99 45 CD : 6F
3F38 1E 20 11 F5 42 CD E5 1F : 57
3F40 C9 2A 99 45 CD 1E 20 11 : ED
3F48 06 43 CD E5 1F C9 2A 9D : AA
3F50 45 CD 1E 20 3A D6 45 B7 : 5C
3F58 28 07 11 2D 43 CD E5 1F : 81
3F60 C9 3A D5 45 B7 28 05 11 : 12
3F68 17 43 18 F1 11 22 43 18 : F1
3F70 EC 2A 9F 45 CD 1E 20 2A : 2F
3F78 8C 45 CD 5B 41 C9 2A A1 : CE
---------------------------------
SUM: 1C 61 60 A1 C2 DF A6 EA 76DA

3F80 45 CD 1E 20 3A B3 45 3C : BE
3F88 CD 9A 41 C9 2A A3 45 CD : 50
3F90 1E 20 2A AF 45 23 CD 5B : A7
3F98 41 C9 2A A5 45 CD 1E 20 : 29
3FA0 3A BF 45 3C CD 9A 41 C9 : EB
3FA8 2A A7 45 CD 1E 20 2A C0 : 0B
3FB0 45 23 CD 5B 41 C9 2A 9D : 61
3FB8 45 CD 1E 20 11 38 43 CD : A9
3FC0 E5 1F CD C4 1F C9 2A 9D : 44
3FC8 45 CD 1E 20 11 43 43 CD : B4
3FD0 E5 1F CD C4 1F C9 2A 9D : 44
3FD8 45 CD 1E 20 3A 93 45 3D : 9F
3FE0 47 CD F1 1F 10 FB C9 CD : C5
3FE8 D6 3F CD 4E 3F CD 1A 3F : 95
3FF0 CD 27 3F CD 8C 3F CD 71 : 09
3FF8 3F C9 11 35 42 CD E5 1F : 61
---------------------------------
SUM: DC 7A 0C F8 D1 3D BE 57 6B7F

4000 11 E8 45 CD E5 1F CD EE : CA
4008 1F C9 11 42 42 CD E5 1F : 4E
4010 11 E8 45 CD E5 1F CD EE : CA
4018 1F C9 CD D6 3F 2A 9D 45 : D6
4020 CD 1E 20 11 4E 42 CD E5 : 5E
4028 1F CD 21 20 F5 CD E7 3F : 15
4030 F1 E6 DF FE 43 CA 46 31 : 38
4038 FE 4E 28 06 FE 53 28 06 : F9
4040 18 D8 AF 3D B7 C9 AF C9 : D4
4048 11 70 42 CD E5 1F CD 21 : 82
4050 20 C9 3E 04 CD A3 1F CD : 87
4058 BC 40 CD DF 38 C9 3E 04 : EB
4060 CD A3 1F CD BC 40 CD FA : 1F
4068 3F CD 09 20 C9 3E 04 CD : 0D
4070 A3 1F CD BC 40 CD 0A 40 : A2
4078 21 00 00 22 70 1F 22 6E : 62
---------------------------------
SUM: 10 61 A1 9F A5 1F 14 CB 9BBA

4080 1F CD EB 38 2A 8A 45 22 : 2A
4088 72 1F CD AF 1F C9 ED 5B : 3D
4090 72 1F 2A 8E 45 ED 52 20 : ED
4098 03 3E 0F C9 2A 84 45 22 : 2E
40A0 70 1F CD A6 1F CD EB 38 : 11
40A8 C9 2A 84 45 22 70 1F 22 : 8F
40B0 6E 1F 2A 8A 45 22 72 1F : 39
40B8 CD AC 1F C9 21 E8 45 3A : E9
40C0 5D 1F 77 23 36 3A 23 ED : 96
40C8 5B 74 1F 06 0D CD DB 40 : E9
40D0 36 2E 23 06 03 CD DB 40 : 78
40D8 36 00 C9 13 1A 77 23 05 : CB
40E0 20 F9 C9 CD 6D 40 DA 1D : 53
40E8 41 CD A9 40 DA 1D 41 AF : DE
40F0 32 90 45 C9 3A 90 45 B7 : 96
40F8 C8 D5 CD 1A 40 D1 D8 C0 : 2D
---------------------------------
SUM: F9 49 91 AE 80 14 BE 27 A3AA

4100 3E 0C CD F4 1F 11 E8 45 : 68
4108 CD E3 40 C9 CD 5E 40 DA : FE
4110 1D 41 CD 8E 40 DA 1D 41 : 31
4118 AF 32 90 45 C9 FE 0F 28 : B4
4120 07 CD 33 20 CD 48 40 C9 : 45
4128 11 7E 42 CD E5 1F CD C4 : 33
4130 1F 18 F1 21 00 00 3E 10 : 97
4138 CB 23 CB 12 ED 6A ED 42 : 51
4140 38 03 1C 18 01 09 3D 20 : D6
4148 EF C9 3E 00 06 08 CB 22 : F1
4150 8F 9B 38 03 14 18 01 83 : 15
4158 10 F4 C9 E5 D5 C5 F5 AF : F0
4160 DD 77 00 01 10 27 EB CD : 44
4168 33 41 7B CD BF 41 01 E8 : A5
4170 03 EB CD 33 41 7B CD BF : 36
4178 41 01 64 00 EB CD 33 41 : D2
---------------------------------
SUM: F3 E7 A2 B1 7F B6 76 90 C8D1

4180 7B CD BF 41 01 0A 00 EB : 3E
4188 CD 33 41 7B CD BF 41 7D : 06
4190 C6 30 CD F4 1F F1 C1 D1 : 59
4198 E1 C9 D5 F5 57 AF DD 77 : CE
41A0 00 1E 64 CD 4A 41 5F 7A : B3
41A8 CD BF 41 53 1E 0A CD 4A : 5F
41B0 41 5F 7A CD BF 41 7B C6 : 28
41B8 30 CD F4 1F F1 D1 C9 C5 : 60
41C0 47 FE 00 20 0E DD 7E 00 : CE
41C8 FE 00 20 07 3E 20 CD F4 : 44
41D0 1F C1 C9 78 C6 30 CD F4 : D8
41D8 1F 3E 01 DD 77 00 C1 C9 : 3C
41E0 21 00 00 1A D6 30 FE 0A : 49
41E8 D0 13 29 D8 44 4D 29 D8 : 76
41F0 29 D8 09 D8 06 00 4F 09 : 40
41F8 30 E9 C9 53 63 72 65 65 : D4
---------------------------------
SUM: FA D3 9A 4A 68 E2 03 00 5FA6

4200 6E 20 45 64 69 74 6F 72 : F5
4208 20 66 6F 72 20 53 2D 4F : 56
4210 53 20 53 57 4F 52 44 0D : 0F
4218 46 45 20 76 65 72 73 69 : D4
4220 6F 6E 20 31 2E 30 30 0D : C9
4228 0D 00 4E 65 77 20 66 69 : 26
4230 6C 65 2E 0D 00 4E 6F 77 : 40
4238 20 6C 6F 61 64 69 6E 67 : FE
4240 20 00 4E 6F 77 20 73 61 : 48
4248 76 69 6E 67 20 00 28 53 : 4F
4250 29 61 76 65 20 2F 20 28 : FC
4258 4E 29 6F 74 20 73 61 76 : C4
4260 65 20 2F 20 28 43 29 61 : C9
4268 6E 63 65 6C 20 3F 20 00 : 21
4270 48 69 74 20 61 6E 79 20 : AD
4278 6B 65 79 2E 20 00 4D 65 : 49
---------------------------------
SUM: C2 6E 54 30 E6 44 F1 C3 3A8C

4280 6D 6F 72 79 20 69 73 20 : E3
4288 66 75 6C 6C 2E 0D 00 28 : 16
4290 45 29 64 69 74 20 2F 20 : 1E
4298 28 4D 29 6F 72 65 20 66 : 6A
42A0 69 6E 64 20 2F 20 28 43 : 15
42A8 29 6F 6D 6D 61 6E 64 00 : A5
42B0 28 52 29 65 70 6C 61 63 : A8
42B8 65 20 2F 20 28 53 29 6B : E3
42C0 69 70 20 2F 20 28 41 29 : DA
42C8 6C 6C 20 2F 20 28 43 29 : DB
42D0 6F 6D 6D 61 6E 64 00 5B : D7
42D8 46 52 45 45 3A 20 20 20 : BC
42E0 20 20 20 5D 00 5B 50 4E 54 : EA
42E8 52 3A 20 20 20 3A 20 20 : 66
42F0 20 20 20 5D 00 5B 4D 41 : A6
42F8 52 4B 3A 20 20 20 3A 20 : 91
---------------------------------
SUM: CD 09 5D 70 DF 21 71 81 D21D

4300 20 20 20 20 5D 00 20 20 : 1D
4308 20 20 20 20 20 20 20 20 : 00
4310 20 20 20 20 20 20 00 4F : 0F
4318 56 45 52 54 59 50 45 20 : 4F
4320 20 00 49 4E 53 45 52 54 : F5
4328 20 20 20 20 00 45 53 43 : 5B
4330 20 20 20 20 20 20 20 00 : E0
4338 4C 49 4E 45 20 4F 56 45 : 32
4340 52 21 00 4D 45 4D 20 4F : C1
4348 56 45 52 21 21 00 43 4F : C1
4350 4D 4D 41 4E 44 20 3E 3E : 09
4358 00 08 20 20 19 09 20 22 : AC
4360 27 28 29 2A 2B 2C 2D 2F : 55
4368 3A 3B 3C 3D 3E 5B 5D 7B : 5F
4370 7E A1 A2 A3 A4 A5 CE 32 : AD
4378 31 33 52 33 E1 32 F0 32 : 1E
---------------------------------
SUM: 67 20 95 A0 3A 5D A9 97 0759

4380 0B 33 1E 33 8B 33 D1 32 : 50
4388 C0 33 D1 33 79 33 82 33 : 58
4390 07 34 0F 34 B1 34 BD 34 : 54
4398 E3 34 12 35 37 35 EC 33 : E9
43A0 BD 35 92 35 FF 32 05 33 : 22
43A8 CE 32 CE 32 CE 32 CE 32 : 00
43B0 CE 32 CE 32 CE 32 CE 32 : 00
43B8 CE 32 4C 33 73 33 CE 32 : 25
43C0 CE 32 CE 32 CE 32 AA 33 : DD
43C8 CE 32 C0 33 CE 32 CE 32 : F3
43D0 CE 32 07 34 37 34 FF 35 : DA
43D8 CE 32 CE 32 CE 32 CE 32 : 00
43E0 CE 32 CE 32 CE 32 CE 32 : 00
43E8 CE 32 CE 32 CE 32 CE 32 : 00
43F0 CE 32 CE 32 CE 32 CE 32 : 00
43F8 CE 32 00 00 00 00 00 00 : 00
---------------------------------
SUM: 4C 29 57 FC 05 F8 1A F7 CCDD
```

リスト2 全機種共通ラインプリントルーチン

```
4400 C5 D5 E5 FD E5 DD 5E 00 : 9C
4408 CD 3C 44 3A 16 30 32 D4 : D3
4410 45 2A D2 45 FD 21 FC 45 : E5
4418 3A A9 45 57 14 15 CA 27 : 99
4420 44 CD 69 30 C3 1D 44 3A : 08
4428 93 45 57 15 CD 69 30 CD : 77
4430 F4 1F 15 C2 2C 44 FD E1 : 38
4438 E1 D1 C1 C9 3A D1 45 67 : F3
4440 3A 92 45 84 3A 91 45 6F : 14
4448 CD 1E 20 C9             : D4
---------------------------------
SUM: C4 96 3B F0 3C 6F 51 FE 17D9
```

リスト3 X1/turbo専用ラインプリントルーチン

```
4400 C5 D5 E5 FD E5 DD 5E 00 : 9C
4408 CD 4C 44 3A 16 30 32 D4 : E3
4410 45 2A D2 45 FD 21 FC 45 : E5
4418 3A A9 45 57 14 15 CA 27 : 99
4420 44 CD 69 30 C3 1D 44 3A : 08
4428 93 45 57 15 CD 69 30 ED : 97
4430 79 CB A0 3A D4 45 ED 79 : 9D
4438 CB E0 CB D8 AF ED 79 CB : 2E
4440 98 03 15 C2 2C 44 FD E1 : C0
4448 E1 D1 C1 C9 3A D1 45 67 : F3
4450 3A 92 45 84 3A 91 45 6F : 14
4458 4D 7C 87 87 84 26 00 6F : F0
4460 29 29 29 3A 5C 1F FE 50 : 7E
4468 20 01 29 06 30 09 44 4D : 1A
4470 C9                      : C9
---------------------------------
SUM: 3E BD 5F 00 CF EF F9 6E 1294
```

リスト4 FE.ASM

```
0000              1  ;======================================
0000              2  ;
0000              3  ;     Screen Editor for S-OS SWORD
0000              4  ;            FE version 1.00
0000              5  ;       Programmed by H.Matsufuji
0000              6  ; Since May 4,1995.
0000              7  ;
0000              8  ;======================================
0000              9
0000             10
1FF4 P           11  _PRINT  EQU     1FF4H
1FF1 P           12  _PRNTS  EQU     1FF1H
1FEE P           13  _LTNL   EQU     1FEEH
1FE5 P           14  _MSX    EQU     1FE5H
1FC4 P           15  _BELL   EQU     1FC4H
1FAF P           16  _WOPEN  EQU     1FAFH
1FAC P           17  _WRD    EQU     1FACH
1FA6 P           18  _RDD    EQU     1FA6H
1FA3 P           19  _FILE   EQU     1FA3H
1F9A P           20  _POKE   EQU     1F9AH
1F97 P           21  _POKE@  EQU     1F97H
1F94 P           22  _PEEK   EQU     1F94H
1F91 P           23  _PEEK@  EQU     1F91H
2009 P           24  _ROPEN  EQU     2009H
201E P           25  _LOC    EQU     201EH
2021 P           26  _FLGET  EQU     2021H
2030 P           27  _WIDCH  EQU     2030H
2033 P           28  _ERROR  EQU     2033H
0000             29  ;
1F76 P           30  _KBFAD  EQU     1F76H
1F74 P           31  _IBFAD  EQU     1F74H
1F72 P           32  _SIZE   EQU     1F72H
1F70 P           33  _DTADR  EQU     1F70H
1F6E P           34  _EXADR  EQU     1F6EH
1F6A P           35  _MEMAX  EQU     1F6AH
1F68 P           36  _WKSIZ  EQU     1F68H
1F5D P           37  _DSK    EQU     1F5DH
1F5C P           38  _WIDTH  EQU     1F5CH
1F5B P           39  _MAXLN  EQU     1F5BH
0000             40
0000             41
0000 P           42  __NULL  EQU     0
0009 P           43  __TAB   EQU     9
000C P           44  __CLS   EQU     12
000D P           45  __CR    EQU     13
001A P           46  __EOF   EQU     26
0020 P           47  __SPACE EQU     32
0000             48
```


▶やっぱりゲームアンソロジーシリーズは光っている。「バラデューク」を見て，ゲーセンの懐かしい興奮が蘇ってきた。まだ17歳なのにおじさんみたいなこといってるなあ。確かに髭も生えてきて汚らしくなってきたけど……。　成木　出哉(17)滋賀県

```
0000                49
4400 P              50  line_print      EQU     4400H
0000                51
0000                52
0000                53          OFFSET  5000H-3000H
3000                54          ORG     3000H
3000                55
3000                56
3000                57  ;--------------------------------------
3000                58  ; Start
3000                59
3000                60
3000                61  cold_start:
3000 C3 E3 30       62          JP      cold0
3003                63
3003                64
3003                65  ;--------------------------------------
3003                66  ; Configrations
3003                67
3003                68
3003 00 48          69  MEMORY_MIN:     DW      04800H
3005                70
3005                71  inital_tab_width:
3005 08             72                  DB      8
3006 14             73  view_offset_w:  DB      20
3007                74
3007 B0             75  chr_TAB:        DB      "-"
3008 A3             76  chr_CR:         DB      ","
3009 20             77  chr_SPACE:      DB      " "
300A A5             78  chr_OTHER:      DB      "."
300B 5B 45 4F 46    79  str_EOF:        DB      "[EOF]",__EOF,0,0
300F 5D 1A 00 00
3013                80
3013 03             81  un_chr_atri:    DB      3       ; unknown color
3014 05             82  EOF_str_atri:   DB      5       ; EOF string color
3015 06             83  sp_chr_atri:    DB      6       ; CR and TAB color
3016 07             84  nm_chr_atri:    DB      7       ; Normal color
3017                85
3017 55 4E 54 49    86  untitle:        DB      "UNTITLED   .   ",0
301B 54 4C 45 44
301F 20 20 20 20
3023 20 2E 20 20
3027 20 00
3029                87
3029                88  key_assign_table:
3029                89  ;                       08 19 2A 3B 4C 5D 6E 7F
3029 00 0C 00 00    90              DB       0,12, 0, 0, 1, 3,11, 0 ; 00
302D 01 03 0B 00
3031 0E 0D 00 00    91              DB      14,13, 0, 0, 0,15, 0, 9 ; 08
3035 00 0F 00 09
3039 00 00 09 02    92              DB       0, 0, 9, 2, 0, 0, 0, 5 ; 10
303D 00 00 00 05
3041 04 00 06 07    93              DB       4, 0, 6, 7, 1, 2, 3, 4 ; 18
3045 01 02 03 04
3049 00 00 00 00    94              DB       0, 0, 0, 0, 0, 0, 0, 0 ; 20
304D 00 00 00 00
3051 00 00 00 00    95              DB       0, 0, 0, 0,24, 0,23, 0 ; 28
3055 18 00 17 00
3059 10 11 12 13    96              DB      16,17,18,19,20,10,22, 0 ; 30
305D 14 0A 16 00
3061 15 08 00 00    97              DB      21, 8, 0, 0, 0, 0, 0, 0 ; 38
3065 00 00 00 00
3069                98
3069                99
3069               100  ;--------------------------------------
3069               101  ; Subroutine for line_print
3069               102
3069               103
3069               104  next_chr:
3069 7B            105          LD      A,E
306A B7            106          OR      A
306B C2 C2 30      107          JP      NZ,nxt_c1
306E 7E            108  nxt_c4: LD      A,(HL)
306F 23            109          INC     HL
3070 FD 23         110          INC     IY
3072 FE 20         111          CP      32
3074 D0            112          RET     NC
3075 FE 09         113          CP      __TAB
3077 CA 92 30      114          JP      Z,nxt_c5
307A FE 0D         115          CP      __CR
307C CA 9E 30      116          JP      Z,nxt_c6
307F B7            117          OR      A
3080 CA AA 30      118          JP      Z,nxt_c7
3083 FE 1A         119          CP      __EOF
3085 CA B6 30      120          JP      Z,nxt_c8
3088 3A 13 30      121          LD      A,(un_chr_atri)
308B 32 D4 45      122          LD      (chr_atri),A
308E 3A 0A 30      123          LD      A,(chr_OTHER)
3091 C9            124          RET
3092               125  ;
3092 1E 01         126  nxt_c5: LD      E,1
3094 3A 15 30      127          LD      A,(sp_chr_atri)
3097 32 D4 45      128          LD      (chr_atri),A
309A 3A 07 30      129          LD      A,(chr_TAB)
309D C9            130          RET
309E               131  ;
309E 1E 02         132  nxt_c6: LD      E,2
30A0 3A 15 30      133          LD      A,(sp_chr_atri)
30A3 32 D4 45      134          LD      (chr_atri),A
30A6 3A 08 30      135          LD      A,(chr_CR)
30A9 C9            136          RET
30AA               137  ;
30AA 21 0B 30      138  nxt_c7: LD      HL,str_EOF
30AD 3A 14 30      139          LD      A,(EOF_str_atri)
30B0 32 D4 45      140          LD      (chr_atri),A
30B3 C3 6E 30      141          JP      nxt_c4
30B6               142  ;
30B6 1E 02         143  nxt_c8: LD      E,2
30B8 3A 16 30      144          LD      A,(nm_chr_atri)
30BB 32 D4 45      145          LD      (chr_atri),A
30BE 3A 09 30      146          LD      A,(chr_SPACE)
30C1 C9            147          RET
30C2               148  ;
30C2 FE 01         149  nxt_c1: CP      1
30C4 CA CB 30      150          JP      Z,nxt_c2
30C7 3A 09 30      151          LD      A,(chr_SPACE)
30CA C9            152          RET
30CB               153  ;
30CB FD 7E 00      154  nxt_c2: LD      A,(IY)
30CE B7            155          OR      A
30CF C2 D8 30      156          JP      NZ,nxt_c3
30D2 FD 23         157          INC     IY
30D4 3A 09 30      158          LD      A,(chr_SPACE)
30D7 C9            159          RET
30D8               160  ;
30D8 1E 00         161  nxt_c3: LD      E,0
30DA 3A 16 30      162          LD      A,(nm_chr_atri)
30DD 32 D4 45      163          LD      (chr_atri),A
30E0 C3 6E 30      164          JP      nxt_c4
30E3               165
30E3               166
30E3               167  ;--------------------------------------
30E3               168  ; Main
30E3               169
30E3               170
30E3 11 FB 41      171  cold0:  LD      DE,startup_mes
30E6 CD E5 1F      172          CALL    _MSX
30E9 DD 21 E4 45   173          LD      IX,local_work
30ED CD 4A 31      174          CALL    mem_workarea_initalize
30F0 CD 65 31      175          CALL    console_workarea_initalize
30F3 CD C4 31      176          CALL    screen_workarea_initalize
30F6 CD E7 31      177          CALL    key_initalize
30F9 CD 03 32      178          CALL    special_workarea_initalize
30FC 3A 05 30      179          LD      A,(inital_tab_width)
30FF 3D            180          DEC     A
3100 4F            181          LD      C,A
3101 CD EF 31      182          CALL    tab_table_initalize
3104 CD DF 38      183          CALL    text_clear
3107               184  ;
3107 ED 5B 76 1F   185          LD      DE,(_KBFAD)
310B 1A            186          LD      A,(DE)
310C FE 23         187          CP      "#"
310E 28 17         188          JR      Z,cold2
3110 1A            189  cold1:  LD      A,(DE)
3111 13            190          INC     DE
3112 B7            191          OR      A
3113 28 12         192          JR      Z,cold2
3115 FE 20         193          CP      " "
3117 20 F7         194          JR      NZ,cold1
3119 CD 5E 40      195          CALL    file_open_r
311C 38 18         196          JR      C,cold3
311E CD 8E 40      197          CALL    file_read
3121 DA 1D 41      198          JP      C,f_err
3124 C3 3D 31      199          JP      cold5
3127               200  ;
3127 11 17 30      201  cold2:  LD      DE,untitle
312A CD 52 40      202          CALL    new_file
312D 11 2A 42      203  cold4:  LD      DE,mes_1
3130 CD E5 1F      204          CALL    _MSX
3133 C3 3D 31      205          JP      cold5
3136               206  ;
3136 FE 08         207  cold3:  CP      8
3138 C2 33 20      208          JP      NZ,_ERROR
313B 18 F0         209          JR      cold4
313D               210  ;
313D CD 28 32      211  cold5:  CALL    pointer_drive
3140 3E 0C         212          LD      A,__CLS
3142 CD F4 1F      213          CALL    _PRINT
3145 C9            214          RET
3146               215
3146               216
3146               217  ;--------------------------------------
3146               218  ; Carry frag opration
3146               219
3146               220
3146               221  scf_ret:
3146 37            222          SCF
3147 C9            223          RET
3148               224
3148               225  rcf_ret:
3148 B7            226          OR      A
3149 C9            227          RET
314A               228
314A               229
314A               230  ;--------------------------------------
314A               231  ; Work area initialize
314A               232
314A               233
314A               234  mem_workarea_initalize:
314A 2A 6A 1F      235          LD      HL,(_MEMAX)
314D 22 82 45      236          LD      (mem_max),HL
3150 22 88 45      237          LD      (text_max),HL
3153 ED 4B 03 30   238          LD      BC,(MEMORY_MIN)
3157 ED 43 80 45   239          LD      (mem_min),BC
315B ED 43 84 45   240          LD      (text_start),BC
315F ED 42         241          SBC     HL,BC
3161 22 8E 45      242          LD      (text_area),HL
3164 C9            243          RET
3165               244
3165               245
3165               246  console_workarea_initalize:
3165 AF            247          XOR     A
3166 32 91 45      248          LD      (console_x),A
3169 32 92 45      249          LD      (console_y),A
316C 32 9B 45      250          LD      (v_fnam_xy),A
316F 32 9D 45      251          LD      (v_mode_xy),A
3172 3A 5C 1F      252          LD      A,(_WIDTH)
3175 32 93 45      253          LD      (console_width),A
3178 D6 07         254          SUB     7
317A 32 A3 45      255          LD      (pointer_line_xy),A
317D 32 A7 45      256          LD      (mark_line_xy),A
3180 D6 04         257          SUB     4
3182 32 A1 45      258          LD      (pointer_x_xy),A
3185 32 A5 45      259          LD      (mark_x_xy),A
3188 D6 06         260          SUB     6
318A 32 97 45      261          LD      (v_pointer_xy),A
318D 32 99 45      262          LD      (v_mark_xy),A
3190 D6 06         263          SUB     6
3192 32 9F 45      264          LD      (free_xy),A
3195 D6 06         265          SUB     6
3197 32 95 45      266          LD      (v_free_xy),A
319A 3A 5B 1F      267          LD      A,(_MAXLN)
319D 3D            268          DEC     A
319E 32 9E 45      269          LD      (v_mode_xy+1),A
31A1 32 96 45      270          LD      (v_free_xy+1),A
31A4 32 98 45      271          LD      (v_pointer_xy+1),A
31A7 32 A2 45      272          LD      (pointer_x_xy+1),A
31AA 32 A4 45      273          LD      (pointer_line_xy+1),A
31AD 32 A0 45      274          LD      (free_xy+1),A
31B0 32 A0 45      275          LD      (free_xy+1),A
31B3 3D            276          DEC     A
31B4 32 94 45      277          LD      (console_length),A
31B7 32 9C 45      278          LD      (v_fnam_xy+1),A
31BA 32 9A 45      279          LD      (v_mark_xy+1),A
31BD 32 A6 45      280          LD      (mark_x_xy+1),A
31C0 32 A8 45      281          LD      (mark_line_xy+1),A
31C3 C9            282          RET
31C4               283
31C4               284
31C4               285  screen_workarea_initalize:
31C4 AF            286          XOR     A
31C5 32 A9 45      287          LD      (view_offset),A
```

▶現在就職活動中だが，いまのところ「コイツだけは絶対許さん！」という方にお会いしてないので，順調といえる。　平　勝久(21)大阪府

```
31C8 32 AE 45       288          LD      (pointer_x),A
31CB 32 B3 45       289          LD      (cursor_x),A
31CE 21 00 00       290          LD      HL,0
31D1 22 AA 45       291          LD      (view_line),HL
31D4 22 AF 45       292          LD      (pointer_line),HL
31D7 2A 84 45       293          LD      HL,(text_start)
31DA 22 AC 45       294          LD      (view_adr),HL
31DD CD 96 39       295          CALL    calc_pointer_address
31E0 CD BD 3C       296          CALL    set_mark_2
31E3 CD 9F 3A       297          CALL    save_cursor_x
31E6 C9             298          RET
31E7                299
31E7                300
31E7                301 key_initalize:
31E7 AF             302          XOR     A
31E8 32 D6 45       303          LD      (escape_mode),A
31EB 32 D5 45       304          LD      (type_mode),A
31EE C9             305          RET
31EF                306
31EF                307
31EF                308 tab_table_initalize:
31EF 21 FC 45       309          LD      HL,tab_info_table
31F2 06 00          310          LD      B,0
31F4 78             311 tti1:    LD      A,B
31F5 A1             312          AND     C
31F6 20 04          313          JR      NZ,tti2
31F8 3E 01          314          LD      A,1
31FA 18 01          315          JR      tti3
31FC AF             316 tti2:    XOR     A
31FD 77             317 tti3:    LD      (HL),A
31FE 23             318          INC     HL
31FF 04             319          INC     B
3200 20 F2          320          JR      NZ,tti1
3202 C9             321          RET
3203                322
3203                323
3203                324 special_workarea_initalize:
3203 CD FA 3D       325          CALL    cant_paste
3206 2A 68 1F       326          LD      HL,(_WKSIZ)
3209 11 00 03       327          LD      DE,300H
320C B7             328          OR      A
320D ED 52          329          SBC     HL,DE
320F 22 DF 45       330          LD      (buffer_size),HL
3212 21 00 00       331          LD      HL,0
3215 22 D7 45       332          LD      (undo_1_b_adr),HL
3218 11 00 01       333          LD      DE,100H
321B 19             334          ADD     HL,DE
321C 22 D9 45       335          LD      (find_b_adr),HL
321F 19             336          ADD     HL,DE
3220 22 DB 45       337          LD      (replace_b_adr),HL
3223 19             338          ADD     HL,DE
3224 22 DD 45       339          LD      (copy_b_adr),HL
3227 C9             340          RET
3228                341
3228                342
3228                343 ;-------------------------------------
3228                344 ; EDITOR OP
3228                345
3228                346
3228                347 pointer_drive:
3228 AF             348          XOR     A
3229 32 CA 45       349          LD      (flag_cmd_mode),A
322C CD 96 3E       350          CALL    view_screen
322F CD 9F 3E       351 pd09:    CALL    view_text
3232                352 ;
3232 CD 3E 3B       353 pd1:     CALL    get_line
3235 CD 4E 39       354 pd11:    CALL    calc_view_line
3238 DD 36 01 00    355          LD      (IX+1),0
323C DC 9F 3E       356          CALL    C,view_text
323F CD 8C 3F       357          CALL    view_pointer_line
3242 CD 2C 3B       358          CALL    calc_locate_y
3245 CD A6 3A       359          CALL    load_cursor_x
3248 CD DC 3A       360          CALL    calc_pointer_x
324B CD AD 3A       361          CALL    calc_cursor_x
324E 18 06          362          JR      pd3
3250 CD AD 3A       363 pd2:     CALL    calc_cursor_x
3253 CD 9F 3A       364          CALL    save_cursor_x
3256 3A CA 45       365 pd3:     LD      A,(flag_cmd_mode)
3259 B7             366          OR      A
325A CC 7E 3F       367          CALL    Z,view_cursor_x
325D CD 0F 3B       368          CALL    calc_locate_x
3260 DC B6 32       369          CALL    C,offset_review
3263 CD DF 3E       370          CALL    view_line_buffer
3266                371 ;
3266 CD C4 32       372          CALL    keyin
3269 FE 20          373          CP      20H
326B DA 94 32       374          JP      C,control_code
326E 4F             375          LD      C,A
326F 3A D6 45       376          LD      A,(escape_mode)
3272 B7             377          OR      A
3273 79             378          LD      A,C
3274 20 13          379          JR      NZ,pd4
3276 3A D5 45       380 type:    LD      A,(type_mode)
3279 B7             381          OR      A
327A 79             382          LD      A,C
327B 20 06          383          JR      NZ,type2
327D CD 14 3D       384          CALL    insert_type
3280 C3 50 32       385          JP      pd2
3283 CD 4C 3D       386 type2:   CALL    over_type
3286 C3 50 32       387          JP      pd2
3289                388 ;
3289 FE 80          389 pd4:     CP      80H
328B D2 CE 32       390          JP      NC,c_null
328E FE 40          391          CP      40H
3290 38 02          392          JR      C,control_code
3292 E6 1F          393          AND     1FH
3294                394
3294                395 ;
3294                396 control_code:
3294 21 29 30       397          LD      HL,key_assign_table
3297 16 00          398          LD      D,0
3299 5F             399          LD      E,A
329A 19             400          ADD     HL,DE
329B 7E             401          LD      A,(HL)
329C F5             402          PUSH    AF
329D 3A CA 45       403          LD      A,(flag_cmd_mode)
32A0 B7             404          OR      A
32A1 28 05          405          JR      Z,c_c1
32A3 21 B8 43       406          LD      HL,control_code_table_cmd
32A6 18 03          407          JR      c_c2
32A8 21 76 43       408 c_c1:    LD      HL,control_code_table
32AB F1             409 c_c2:    POP     AF
32AC CB 27          410          SLA     A
32AE 5F             411          LD      E,A
32AF 19             412          ADD     HL,DE
32B0 5E             413          LD      E,(HL)
32B1 23             414          INC     HL
32B2 56             415          LD      D,(HL)
32B3 62             416          LD      H,D
32B4 6B             417          LD      L,E
32B5 E9             418          JP      (HL)
32B6                419
32B6                420
32B6                421 offset_review:
32B6 CD 13 39       422          CALL    calc_view_offset
32B9 DD 36 01 01    423          LD      (IX+1),1
32BD CD 9F 3E       424          CALL    view_text
32C0 CD 0F 3B       425          CALL    calc_locate_x
32C3 C9             426          RET
32C4                427
32C4                428
32C4                429 keyin:
32C4 2A B5 45       430          LD      HL,(locate_xy)
32C7 CD 1E 20       431          CALL    _LOC
32CA CD 21 20       432          CALL    _FLGET
32CD C9             433          RET
32CE                434
32CE                435
32CE                436 c_null:
32CE                437 c_reserved:
32CE C3 50 32       438          JP      pd2
32D1                439
32D1                440
32D1                441 c_quit:
32D1 CD 83 3C       442          CALL    exit_escape_mode
32D4 CD 91 3B       443          CALL    put_line
32D7 DA 50 32       444          JP      C,pd2
32DA CD F4 40       445          CALL    close_file
32DD DA 50 32       446          JP      C,pd2
32E0 C9             447          RET
32E1                448
32E1                449
32E1                450 c_up:
32E1 CD 91 3B       451          CALL    put_line
32E4 DA 50 32       452          JP      C,pd2
32E7 CD 5A 3A       453          CALL    pointer_line_dec
32EA DA 43 33       454          JP      C,c_r2
32ED C3 32 32       455          JP      pd1
32F0                456
32F0                457
32F0                458 c_down:
32F0 CD 91 3B       459          CALL    put_line
32F3 DA 50 32       460          JP      C,pd2
32F6 CD 47 3A       461          CALL    pointer_line_inc
32F9 DA 64 33       462          JP      C,c_l2
32FC C3 32 32       463          JP      pd1
32FF                464
32FF                465
32FF                466 c_cursor_top:
32FF CD 52 38       467          CALL    top_cursor
3302 C3 50 32       468          JP      pd2
3305                469
3305                470
3305                471 c_cursor_botom:
3305 CD 5A 38       472          CALL    botom_cursor
3308 C3 50 32       473          JP      pd2
330B                474
330B                475
330B                476 c_up_page:
330B CD 91 3B       477          CALL    put_line
330E DA 50 32       478          JP      C,pd2
3311 3A 94 45       479          LD      A,(console_length)
3314 47             480          LD      B,A
3315 CD 5A 3A       481 up151:   CALL    pointer_line_dec
3318 05             482          DEC     B
3319 20 FA          483          JR      NZ,up151
331B C3 32 32       484          JP      pd1
331E                485
331E                486
331E                487 c_down_page:
331E CD 91 3B       488          CALL    put_line
3321 DA 50 32       489          JP      C,pd2
3324 3A 94 45       490          LD      A,(console_length)
3327 47             491          LD      B,A
3328 CD 47 3A       492 down151:CALL    pointer_line_inc
332B 05             493          DEC     B
332C 20 FA          494          JR      NZ,down151
332E C3 32 32       495          JP      pd1
3331                496
3331                497
3331                498 c_right:
3331 CD BE 39       499          CALL    pointer_x_inc
3334 D2 50 32       500          JP      NC,pd2
3337 CD 91 3B       501 c_r1:    CALL    put_line
333A DA 50 32       502          JP      C,pd2
333D CD 47 3A       503          CALL    pointer_line_inc
3340 DA 50 32       504          JP      C,pd2
3343 CD 52 38       505 c_r2:    CALL    top_cursor
3346 CD 9F 3A       506          CALL    save_cursor_x
3349 C3 32 32       507          JP      pd1
334C                508
334C                509
334C                510 c_right_cmd:
334C CD BE 39       511          CALL    pointer_x_inc
334F C3 50 32       512          JP      pd2
3352                513
3352                514
3352                515 c_left:
3352 CD D1 39       516          CALL    pointer_x_dec
3355 D2 50 32       517          JP      NC,pd2
3358 CD 91 3B       518 c_l1:    CALL    put_line
335B DA 50 32       519          JP      C,pd2
335E CD 5A 3A       520          CALL    pointer_line_dec
3361 DA 32 32       521          JP      C,pd1
3364 CD 3E 3B       522 c_l2:    CALL    get_line
3367 CD 5A 38       523          CALL    botom_cursor
336A CD AD 3A       524          CALL    calc_cursor_x
336D CD 9F 3A       525          CALL    save_cursor_x
3370 C3 35 32       526          JP      pd11
3373                527
3373                528
3373                529 c_left_cmd:
3373 CD D1 39       530          CALL    pointer_x_dec
3376 C3 50 32       531          JP      pd2
3379                532
3379                533
3379                534 c_skip_foward:
3379 CD DD 39       535          CALL    word_skip_foward
337C D2 50 32       536          JP      NC,pd2
337F C3 37 33       537          JP      c_r1
3382                538
3382                539
```


▶正妻の座(6月号107ページ参照)は揺るぎませんが,動かないゲームがあったり,ハングしたりする度に愛人にかかりきりになってしまいます。これだからマニアは……。
八木 明(29)神奈川県

```
3382                540 c_skip_back:
3382 CD 0D 3A       541         CALL    word_skip_back
3385 D2 50 32       542         JP      NC,pd2
3388 C3 58 33       543         JP      c_11
338B                544
338B                545
338B                546 c_escape:
338B CD 91 3B       547         CALL    put_line
338E DA 50 32       548         JP      C,pd2
3391 3A D6 45       549         LD      A,(escape_mode)
3394 B7             550         OR      A
3395 28 06          551         JR      Z,c_escl
3397 CD 83 3C       552         CALL    exit_escape_mode
339A C3 32 32       553         JP      pd1
339D 3D             554 c_escl: DEC     A
339E 32 D6 45       555         LD      (escape_mode),A
33A1 CD 96 3C       556         CALL    set_mark
33A4 CD 4E 3F       557         CALL    view_type_mode
33A7 C3 32 32       558         JP      pd1
33AA                559
33AA                560
33AA                561 c_escape_cmd:
33AA 3A D6 45       562         LD      A,(escape_mode)
33AD B7             563         OR      A
33AE 28 06          564         JR      Z,c_ecl
33B0 CD 83 3C       565         CALL    exit_escape_mode
33B3 C3 50 32       566         JP      pd2
33B6 3D             567 c_ecl:  DEC     A
33B7 32 D6 45       568         LD      (escape_mode),A
33BA CD 4E 3F       569         CALL    view_type_mode
33BD C3 50 32       570         JP      pd2
33C0                571
33C0                572
33C0                573 c_change_typemode:
33C0 3A D5 45       574         LD      A,(type_mode)
33C3 EE 01          575         XOR     1
33C5 32 D5 45       576         LD      (type_mode),A
33C8 CD 83 3C       577         CALL    exit_escape_mode
33CB CD 4E 3F       578         CALL    view_type_mode
33CE C3 50 32       579         JP      pd2
33D1                580
33D1                581
33D1                582 c_tab_hid:
33D1 CD 83 3C       583         CALL    exit_escape_mode
33D4 3A 07 30       584         LD      A,(chr_TAB)
33D7 47             585         LD      B,A
33D8 3A 5A 43       586         LD      A,(chr_TAB_swap)
33DB 32 07 30       587         LD      (chr_TAB),A
33DE 78             588         LD      A,B
33DF 32 5A 43       589         LD      (chr_TAB_swap),A
33E2 DD 36 01 01    590         LD      (IX+1),1
33E6 CD 9F 3E       591         CALL    view_text
33E9 C3 35 32       592         JP      pd11
33EC                593
33EC                594
33EC                595 c_cr_hid:
33EC CD 83 3C       596         CALL    exit_escape_mode
33EF 3A 08 30       597         LD      A,(chr_CR)
33F2 47             598         LD      B,A
33F3 3A 5B 43       599         LD      A,(chr_CR_swap)
33F6 32 08 30       600         LD      (chr_CR),A
33F9 78             601         LD      A,B
33FA 32 5B 43       602         LD      (chr_CR_swap),A
33FD DD 36 01 01    603         LD      (IX+1),1
3401 CD 9F 3E       604         CALL    view_text
3404 C3 35 32       605         JP      pd11
3407                606
3407                607
3407                608 c_tab:
3407 CD 8E 3C       609         CALL    delete_mark
340A 0E 09          610         LD      C,__TAB
340C C3 76 32       611         JP      type
340F                612
340F                613
340F                614 c_back_space:
340F CD 8E 3C       615         CALL    delete_mark
3412 3A AE 45       616         LD      A,(pointer_x)
3415 B7             617         OR      A
3416 CA 59 34       618         JP      Z,c_delete_cr
3419 CD 78 3C       619         CALL    calc_pointer_inline
341C 54             620         LD      D,H
341D 5D             621         LD      E,L
341E 1B             622         DEC     DE
341F 7E             623 c_bs0:  LD      A,(HL)
3420 12             624         LD      (DE),A
3421 FE 0D          625         CP      __CR
3423 CA 2E 34       626         JP      Z,c_bs1
3426 B7             627         OR      A
3427 CA 2E 34       628         JP      Z,c_bs1
342A 23             629         INC     HL
342B 13             630         INC     DE
342C 18 F1          631         JR      c_bs0
342E                632 ;
342E CD D1 39       633 c_bs1:  CALL    pointer_x_dec
3431 CD 08 3D       634         CALL    edited_line
3434 C3 50 32       635         JP      pd2
3437                636
3437                637
3437                638 c_back_space_cmd:
3437 3A AE 45       639         LD      A,(pointer_x)
343A B7             640         OR      A
343B CA 50 32       641         JP      Z,pd2
343E CD 78 3C       642         CALL    calc_pointer_inline
3441 54             643         LD      D,H
3442 5D             644         LD      E,L
3443 1B             645         DEC     DE
3444 7E             646 c_bsc0: LD      A,(HL)
3445 12             647         LD      (DE),A
3446 FE 0D          648         CP      __CR
3448 CA 53 34       649         JP      Z,c_bsc1
344B B7             650         OR      A
344C CA 53 34       651         JP      Z,c_bsc1
344F 23             652         INC     HL
3450 13             653         INC     DE
3451 18 F1          654         JR      c_bsc0
3453 CD D1 39       655 c_bsc1: CALL    pointer_x_dec
3456 C3 50 32       656         JP      pd2
3459                657
3459                658
3459                659 ; Delete CR.
3459                660 ;
3459                661 c_delete_cr:
3459 CD 8E 3C       662         CALL    delete_mark
345C CD 91 3B       663         CALL    put_line
345F DA 50 32       664         JP      C,pd2
3462 CD 5A 3A       665         CALL    pointer_line_dec
3465 DA 32 32       666         JP      C,pd1
3468 CD 3E 3B       667         CALL    get_line
346B AF             668         XOR     A
346C 3D             669         DEC     A
346D 32 B3 45       670         LD      (cursor_x),A
3470 CD DC 3A       671         CALL    calc_pointer_x
3473 CD AD 3A       672         CALL    calc_cursor_x
3476 CD 9F 3A       673         CALL    save_cursor_x
3479 2A B1 45       674         LD      HL,(pointer_adr)
347C CD 71 3A       675         CALL    next_line
347F 11 01 00       676         LD      DE,1
3482 CD 0D 3C       677         CALL    move_text_minus
3485 CD 05 39       678 c_dcl:  CALL    calc_text_free
3488 C3 2F 32       679         JP      pd09
348B                680
348B                681
348B                682 ; Insert CR
348B                683 ;
348B                684 c_insert_cr:
348B CD 8E 3C       685         CALL    delete_mark
348E CD 91 3B       686         CALL    put_line
3491 DA 50 32       687         JP      C,pd2
3494 CD D4 38       688         CALL    calc_pointer
3497 E5             689         PUSH    HL
3498 11 01 00       690         LD      DE,1
349B CD DA 3B       691         CALL    move_text_plus
349E E1             692         POP     HL
349F DA 50 32       693         JP      C,pd2
34A2 36 0D          694         LD      (HL),__CR
34A4 CD 3E 3B       695         CALL    get_line
34A7 DD 36 01 00    696         LD      (IX+1),0
34AB CD 9F 3E       697         CALL    view_text
34AE C3 31 33       698         JP      c_right
34B1                699
34B1                700
34B1                701 ; CR
34B1                702 ;
34B1                703 c_cr:
34B1 3A D5 45       704         LD      A,(type_mode)
34B4 B7             705         OR      A
34B5 28 D4          706         JR      Z,c_insert_cr
34B7 CD 5A 38       707 c_crl:  CALL    botom_cursor
34BA C3 31 33       708         JP      c_right
34BD                709
34BD                710
34BD                711 ; Block delete
34BD                712 ;
34BD                713 c_block_delete:
34BD CD 91 3B       714         CALL    put_line
34C0 DA 50 32       715         JP      C,pd2
34C3 CD BD 3C       716         CALL    set_mark_2
34C6 CD 6C 35       717         CALL    calc_block_length
34C9 DA 50 32       718         JP      C,pd2
34CC 19             719         ADD     HL,DE
34CD CD 0D 3C       720         CALL    move_text_minus
34D0 CD EF 3C       721         CALL    get_mark_2
34D3 CD 9F 3A       722         CALL    save_cursor_x
34D6 DD 36 01 60    723         LD      (IX+1),0
34DA CD 9F 3E       724         CALL    view_text
34DD CD 83 3C       725         CALL    exit_escape_mode
34E0 C3 32 32       726         JP      pd1
34E3                727
34E3                728
34E3                729 ; Block cut
34E3                730 ;
34E3                731 c_block_cut:
34E3 CD 91 3B       732         CALL    put_line
34E6 DA 50 32       733         JP      C,pd2
34E9 CD BD 3C       734         CALL    set_mark_2
34EC CD 6C 35       735         CALL    calc_block_length
34EF DA 50 32       736         JP      C,pd2
34F2 CD B5 3D       737         CALL    save_copy
34F5 DA 50 32       738         JP      C,pd2
34F8 19             739         ADD     HL,DE
34F9 CD 0D 3C       740         CALL    move_text_minus
34FC CD EF 3C       741         CALL    get_mark_2
34FF CD 9F 3A       742         CALL    save_cursor_x
3502 DD 36 01 00    743         LD      (IX+1),0
3506 CD 9F 3E       744         CALL    view_text
3509 CD 83 3C       745         CALL    exit_escape_mode
350C CD F5 3D       746         CALL    can_paste
350F C3 32 32       747         JP      pd1
3512                748
3512                749
3512                750 ; Block copy
3512                751 ;
3512                752 c_block_copy:
3512 CD 91 3B       753         CALL    put_line
3515 DA 50 32       754         JP      C,pd2
3518 CD BD 3C       755         CALL    set_mark_2
351B CD 6C 35       756         CALL    calc_block_length
351E DA 50 32       757         JP      C,pd2
3521 CD B5 3D       758         CALL    save_copy
3524 DA 50 32       759         JP      C,pd2
3527 DD 36 01 00    760         LD      (IX+1),0
352B CD 9F 3E       761         CALL    view_text
352E CD 83 3C       762         CALL    exit_escape_mode
3531 CD F5 3D       763         CALL    can_paste
3534 C3 32 32       764         JP      pd1
3537                765
3537                766
3537                767 ; Block paste
3537                768 ;
3537                769 c_block_paste:
3537 CD 91 3B       770         CALL    put_line
353A DA 50 32       771         JP      C,pd2
353D 3A CD 45       772         LD      A,(flag_cant_past)
3540 B7             773         OR      A
3541 C2 50 32       774         JP      NZ,pd2
3544 CD 83 3C       775         CALL    exit_escape_mode
3547 CD D4 38       776         CALL    calc_pointer
354A E5             777         PUSH    HL
354B CD D2 3D       778         CALL    load_copy
354E D1             779         POP     DE
354F DA 50 32       780         JP      C,pd2
3552 2A CB 45       781         LD      HL,(copy_length)
3555 19             782         ADD     HL,DE
3556 CD 60 3D       783         CALL    calc_pointer_line_x_add
3559 CD 3E 3B       784         CALL    get_line
355C CD AD 3A       785         CALL    calc_cursor_x
355F CD 9F 3A       786         CALL    save_cursor_x
3562 DD 36 01 00    787         LD      (IX+1),0
3566 CD 9F 3E       788         CALL    view_text
3569 C3 35 32       789         JP      pd11
356C                790
356C                791
```

▶モデムを買いました。メモリも4Mバイトになりました。しかし，スロットが２つとも塞がってしまいました(MIDI&メモリボード)。ぬおお，SCSIが使えん。

清水　邦彦(20)富山県

```
356C                792  ; Block length calcuration
356C                793  ;
356C                794  calc_block_length:
356C CD 00 3E       795          CALL    swap_mark_pointer
356F 2A C8 45       796          LD      HL,(mark_2_adr)
3572 3A C4 45       797          LD      A,(mark_2_x)
3575 16 00          798          LD      D,0
3577 5F             799          LD      E,A
3578 19             800          ADD     HL,DE
3579 44             801          LD      B,H
357A 4D             802          LD      C,L
357B 2A C2 45       803          LD      HL,(mark_1_adr)
357E 3A BE 45       804          LD      A,(mark_1_x)
3581 16 00          805          LD      D,0
3583 5F             806          LD      E,A
3584 19             807          ADD     HL,DE
3585 50             808          LD      D,B
3586 59             809          LD      E,C
3587 B7             810          OR      A
3588 ED 52          811          SBC     HL,DE
358A CA 46 31       812          JP      Z,scf_ret
358D 54             813          LD      D,H
358E 5D             814          LD      E,L
358F 60             815          LD      H,B
3590 69             816          LD      L,C
3591 C9             817          RET
3592                818
3592                819
3592                820  c_tab_width_change:
3592 CD 91 3B       821          CALL    put_line
3595 DA 50 32       822          JP      C,pd2
3598 CD 83 3C       823          CALL    exit_escape_mode
359B 3A 59 43       824          LD      A,(tab_width)
359E FE 08          825          CP      8
35A0 20 04          826          JR      NZ,t_wc1
35A2 3E 04          827          LD      A,4
35A4 18 02          828          JR      t_wc2
35A6 3E 08          829  t_wc1:  LD      A,8
35A8 32 59 43       830  t_wc2:  LD      (tab_width),A
35AB 3D             831          DEC     A
35AC 4F             832          LD      C,A
35AD CD EF 31       833          CALL    tab_table_initalize
35B0 DD 36 01 00    834          LD      (IX+1),0
35B4 CD 9F 3E       835          CALL    view_text
35B7 CD 3E 3B       836          CALL    get_line
35BA C3 50 32       837          JP      pd2
35BD                838
35BD                839
35BD                840  c_command_mode:
35BD CD 91 3B       841          CALL    put_line
35C0 DA 50 32       842          JP      C,pd2
35C3 CD 83 3C       843          CALL    exit_escape_mode
35C6 CD BD 3C       844          CALL    set_mark_2
35C9 CD D6 3F       845          CALL    clr_botom
35CC AF             846  c_cmde: XOR     A
35CD 3D             847          DEC     A
35CE 32 CA 45       848          LD      (flag_cmd_mode),A
35D1 3A 91 45       849          LD      A,(console_x)
35D4 C6 0A          850          ADD     A,10
35D6 32 91 45       851          LD      (console_x),A
35D9 3A 93 45       852          LD      A,(console_width)
35DC D6 0A          853          SUB     10
35DE 32 93 45       854          LD      (console_width),A
35E1 3A 9E 45       855          LD      A,(v_mode_xy+1)
35E4 32 B6 45       856          LD      (locate_xy+1),A
35E7 2A 9D 45       857          LD      HL,(v_mode_xy)
35EA CD 1E 20       858          CALL    _LOC
35ED 11 4E 43       859          LD      DE,s_command
35F0 CD E5 1F       860          CALL    _MSX
35F3 3E 0D          861          LD      A,__CR
35F5 32 FC 46       862          LD      (line_buffer),A
35F8 AF             863          XOR     A
35F9 32 AE 45       864          LD      (pointer_x),A
35FC C3 50 32       865          JP      pd2
35FF                866
35FF                867
35FF                868  ;-------------------------------------
35FF                869  ; Command execute
35FF                870
35FF                871
35FF                872  c_cr_cmd:
35FF AF             873          XOR     A
3600 32 CA 45       874          LD      (flag_cmd_mode),A
3603 CD E7 3F       875          CALL    view_botom
3606 3A 91 45       876          LD      A,(console_x)
3609 D6 0A          877          SUB     10
360B 32 91 45       878          LD      (console_x),A
360E 3A 93 45       879          LD      A,(console_width)
3611 C6 0A          880          ADD     A,10
3613 32 93 45       881          LD      (console_width),A
3616 11 FC 46       882          LD      DE,line_buffer
3619 CD 4A 36       883  cccl:   CALL    skip_space
361C 13             884          INC     DE
361D E6 DF          885          AND     0DFH
361F FE 45          886          CP      "E"
3621 CA 5E 36       887          JP      Z,x_edit
3624 FE 51          888          CP      "Q"
3626 CA 93 36       889          JP      Z,x_quit
3629 FE 4C          890          CP      "L"
362B CA 9A 36       891          JP      Z,x_load
362E FE 53          892          CP      "S"
3630 CA C8 36       893          JP      Z,x_save
3633 FE 4E          894          CP      "N"
3635 CA DE 36       895          JP      Z,x_new
3638 FE 42          896          CP      "B"
363A CA 7B 36       897          JP      Z,x_first
363D FE 46          898          CP      "F"
363F CA 03 37       899          JP      Z,x_find
3642 FE 43          900          CP      "C"
3644 CA 72 37       901          JP      Z,x_replace
3647                902  ;       CP      "W"
3647                903  ;       JP      Z,x_width
3647 C3 CC 35       904          JP      c_cmde
364A                905
364A                906
364A                907  skip_space:
364A 1A             908          LD      A,(DE)
364B FE 20          909          CP      __SPACE
364D C0             910          RET     NZ
364E FE 09          911          CP      __TAB
3650 C0             912          RET     NZ
3651 13             913          INC     DE
3652 18 F6          914          JR      skip_space
3654                915
3654                916
3654                917  skip_not_space:
3654 1A             918          LD      A,(DE)
3655 FE 20          919          CP      __SPACE
3657 C8             920          RET     Z
3658 FE 09          921          CP      __TAB
365A C8             922          RET     Z
365B 13             923          INC     DE
365C 18 F6          924          JR      skip_not_space
365E                925
365E                926
365E                927  x_edit:
365E CD 4A 36       928          CALL    skip_space
3661 FE 0D          929          CP      __CR
3663 28 0D          930          JR      Z,x_ed1
3665 CD E0 41       931          CALL    get_number
3668 2B             932          DEC     HL
3669 22 AF 45       933          LD      (pointer_line),HL
366C CD 96 39       934          CALL    calc_pointer_address
366F CD BD 3C       935  x_ed2:  CALL    set_mark_2
3672 CD EF 3C       936  x_ed1:  CALL    get_mark_2
3675 CD 9F 3A       937          CALL    save_cursor_x
3678 C3 32 32       938          JP      pd1
367B                939
367B                940
367B                941  ;x_width:
367B                942  ;       LD      A,(_WIDTH)
367B                943  ;       CP      40
367B                944  ;       JR      Z,x_w1
367B                945  ;       LD      A,40
367B                946  ;x_w0:  CALL    _WIDCH
367B                947  ;       CALL    console_workarea_initalize
367B                948  ;       CALL    view_colums
367B                949  ;       LD      (IX+1),0
367B                950  ;       CALL    view_text
367B                951  ;       JP      c_cmde
367B                952  ;x_w1:  LD      A,80
367B                953  ;       JR      x_w0
367B                954
367B                955
367B                956  x_first:
367B D5             957          PUSH    DE
367C AF             958          XOR     A
367D 32 AE 45       959          LD      (pointer_x),A
3680 32 B3 45       960          LD      (cursor_x),A
3683 21 00 00       961          LD      HL,0
3686 22 AF 45       962          LD      (pointer_line),HL
3689 CD 96 39       963          CALL    calc_pointer_address
368C CD BD 3C       964          CALL    set_mark_2
368F D1             965          POP     DE
3690 C3 19 36       966          JP      cccl
3693                967
3693                968
3693                969  x_quit:
3693 CD F4 40       970          CALL    close_file
3696 DA CC 35       971          JP      C,c_cmde
3699 C9             972          RET
369A                973
369A                974
369A                975  x_load:
369A D5             976          PUSH    DE
369B CD F4 40       977          CALL    close_file
369E D1             978          POP     DE
369F DA CC 35       979          JP      C,c_cmde
36A2 CD 4A 36       980          CALL    skip_space
36A5 3E 0C          981          LD      A,__CLS
36A7 CD F4 1F       982          CALL    _PRINT
36AA CD 0C 41       983          CALL    load_file
36AD DA C0 36       984          JP      C,x_ld1
36B0 CD C4 31       985          CALL    screen_workarea_initalize
36B3 CD 96 3E       986  x_ld0:  CALL    view_screen
36B6 DD 36 01 00    987          LD      (IX+1),0
36BA CD 9F 3E       988          CALL    view_text
36BD C3 CC 35       989          JP      c_cmde
36C0 CD 33 20       990  x_ld1:  CALL    _ERROR
36C3 CD 48 40       991          CALL    pause
36C6 18 EB          992          JR      x_ld0
36C8                993
36C8                994
36C8                995  x_save:
36C8 CD 4A 36       996          CALL    skip_space
36CB 1A             997          LD      A,(DE)
36CC FE 0D          998          CP      __CR
36CE 20 03          999          JR      NZ,x_s1
36D0 11 E8 45      1000          LD      DE,filename
36D3 3E 0C         1001  x_s1:   LD      A,__CLS
36D5 CD F4 1F      1002          CALL    _PRINT
36D8 CD E3 40      1003          CALL    save_file
36DB C3 B3 36      1004          JP      x_ld0
36DE               1005
36DE               1006
36DE               1007  x_new:
36DE D5            1008          PUSH    DE
36DF CD F4 40      1009          CALL    close_file
36E2 D1            1010          POP     DE
36E3 CD 4A 36      1011          CALL    skip_space
36E6 FE 0D         1012          CP      __CR
36E8 28 0C         1013          JR      Z,x_n1
36EA CD 52 40      1014  x_n0:   CALL    new_file
36ED DA FB 36      1015          JP      C,x_n2
36F0 CD C4 31      1016          CALL    screen_workarea_initalize
36F3 C3 B3 36      1017          JP      x_ld0
36F6 11 17 30      1018  x_n1:   LD      DE,untitle
36F9 18 EF         1019          JR      x_n0
36FB 3E 0C         1020  x_n2:   LD      A,__CLS
36FD CD F4 1F      1021          CALL    _PRINT
3700 C3 C0 36      1022          JP      x_ld1
3703               1023
3703               1024
3703               1025  ; Find
3703               1026  ;
3703               1027  x_find:
3703 CD 4A 36      1028          CALL    skip_space
3706 FE 0D         1029          CP      __CR
3708 CA CC 35      1030          JP      Z,c_cmde
370B 47            1031          LD      B,A
370C E6 DF         1032          AND     0DFH
370E FE 4D         1033          CP      "M"
3710 28 1E         1034          JR      Z,x_fn0
3712 78            1035          LD      A,B
3713 13            1036          INC     DE
3714 2A D9 45      1037          LD      HL,(find_b_adr)
3717 1A            1038          LD      A,(DE)
3718 B8            1039          CP      B
3719 CA CC 35      1040          JP      Z,c_cmde
371C               1041  ;
371C 1A            1042  x_fn1:  LD      A,(DE)
371D FE 0D         1043          CP      __CR
```

▶X68000ほど長く使えるPCはないのに数で負けているのは残念である。
藤井　聡志(30)埼玉県

```
371F CA CC 35      1044           JP      Z,c_cmde
3722 B8            1045           CP      B
3723 28 07         1046           JR      Z,x_fn2
3725 CD 9A 1F      1047           CALL    _POKE
3728 23            1048           INC     HL
3729 13            1049           INC     DE
372A 18 F0         1050           JR      x_fn1
372C AF            1051   x_fn2:  XOR     A
372D CD 9A 1F      1052           CALL    _POKE
3730               1053   ;
3730 CD 63 38      1054   x_fn0:  CALL    find_str
3733 DA CC 35      1055           JP      C,c_cmde
3736 CD 38 38      1056           CALL    calc_flash_locate
3739 CD D6 3F      1057           CALL    clr_botom
373C 2A 9D 45      1058           LD      HL,(v_mode_xy)
373F CD 1E 20      1059           CALL    _LOC
3742 11 8F 42      1060           LD      DE,mes_7
3745 CD E5 1F      1061           CALL    _MSX
3748 2A B5 45      1062           LD      HL,(locate_xy)
374B CD 1E 20      1063           CALL    _LOC
374E CD 21 20      1064   x_fn4:  CALL    _FLGET
3751 E6 DF         1065           AND     0DFH
3753 FE 45         1066           CP      "E"
3755 CA 63 37      1067           JP      Z,x_fn5
3758 FE 4D         1068           CP      "M"
375A 28 0D         1069           JR      Z,x_fn6
375C FE 43         1070           CP      "C"
375E CA CC 35      1071           JP      Z,c_cmde
3761 18 EB         1072           JR      x_fn4
3763               1073   ;
3763 CD E7 3F      1074   x_fn5:  CALL    view_botom
3766 C3 6F 36      1075           JP      x_ed2
3769               1076   ;
3769 CD D4 38      1077   x_fn6:  CALL    calc_pointer
376C 23            1078           INC     HL
376D CD 60 3D      1079           CALL    calc_pointer_line_x_add
3770 18 BE         1080           JR      x_fn0
3772               1081   ;
3772               1082   ;
3772               1083   ; Find and replace
3772               1084   ;
3772               1085   x_replace:
3772 AF            1086           XOR     A
3773 32 E1 45      1087           LD      (flag_replace_all),A
3776 CD 4A 36      1088           CALL    skip_space
3779 FE 0D         1089           CP      __CR
377B CA CC 35      1090           JP      Z,c_cmde
377E 47            1091           LD      B,A
377F 13            1092           INC     DE
3780 2A D9 45      1093           LD      HL,(find_b_adr)
3783 1A            1094           LD      A,(DE)
3784 B8            1095           CP      B
3785 CA CC 35      1096           JP      Z,c_cmde
3788 0E 00         1097           LD      C,0
378A               1098   ;
378A 1A            1099   x_lp1:  LD      A,(DE)
378B FE 0D         1100           CP      __CR
378D CA CC 35      1101           JP      Z,c_cmde
3790 B8            1102           CP      B
3791 28 08         1103           JR      Z,x_lp2
3793 CD 9A 1F      1104           CALL    _POKE
3796 0C            1105           INC     C
3797 23            1106           INC     HL
3798 13            1107           INC     DE
3799 18 EF         1108           JR      x_lp1
379B AF            1109   x_lp2:  XOR     A
379C CD 9A 1F      1110           CALL    _POKE
379F 79            1111           LD      A,C
37A0 32 E2 45      1112           LD      (find_str_len),A
37A3               1113   ;
37A3 2A DB 45      1114           LD      HL,(replace_b_adr)
37A6 13            1115           INC     DE
37A7 0E 00         1116           LD      C,0
37A9 1A            1117   x_lp6:  LD      A,(DE)
37AA FE 0D         1118           CP      __CR
37AC CA CC 35      1119           JP      Z,c_cmde
37AF B8            1120           CP      B
37B0 28 08         1121           JR      Z,x_lp7
37B2 CD 9A 1F      1122           CALL    _POKE
37B5 0C            1123           INC     C
37B6 23            1124           INC     HL
37B7 13            1125           INC     DE
37B8 18 EF         1126           JR      x_lp6
37BA AF            1127   x_lp7:  XOR     A
37BB CD 9A 1F      1128           CALL    _POKE
37BE 79            1129           LD      A,C
37BF 32 E3 45      1130           LD      (replace_str_len),A
37C2               1131   ;
37C2 CD 63 38      1132   x_lp0:  CALL    find_str
37C5 DA 2E 38      1133           JP      C,x_lp9
37C8 3A E1 45      1134           LD      A,(flag_replace_all)
37CB B7            1135           OR      A
37CC 20 32         1136           JR      NZ,x_lp5
37CE CD 38 38      1137           CALL    calc_flash_locate
37D1 CD D6 3F      1138           CALL    clr_botom
37D4 2A 9D 45      1139           LD      HL,(v_mode_xy)
37D7 CD 1E 20      1140           CALL    _LOC
37DA 11 B0 42      1141           LD      DE,mes_8
37DD CD E5 1F      1142           CALL    _MSX
37E0 2A B5 45      1143           LD      HL,(locate_xy)
37E3 CD 1E 20      1144           CALL    _LOC
37E6 CD 21 20      1145   x_lp4:  CALL    _FLGET
37E9 E6 DF         1146           AND     0DFH
37EB FE 52         1147           CP      "R"
37ED CA 00 38      1148           JP      Z,x_lp5
37F0 FE 53         1149           CP      "S"
37F2 28 22         1150           JR      Z,x_lp55
37F4 FE 43         1151           CP      "C"
37F6 CA CC 35      1152           JP      Z,c_cmde
37F9 FE 41         1153           CP      "A"
37FB CA 20 38      1154           JP      Z,x_lp8
37FE 18 C2         1155           JR      x_lp0
3800               1156   ;
3800 CD 94 38      1157   x_lp5:  CALL    replace
3803 DA 2E 38      1158           JP      C,x_lp9
3806 CD D4 38      1159           CALL    calc_pointer
3809 3A E3 45      1160           LD      A,(replace_str_len)
380C 16 00         1161           LD      D,0
380E 5F            1162           LD      E,A
380F 19            1163           ADD     HL,DE
3810 CD 60 3D      1164           CALL    calc_pointer_line_x_add
3813 C3 C2 37      1165           JP      x_lp0
3816               1166   ;
3816 CD D4 38      1167   x_lp55: CALL    calc_pointer
3819 23            1168           INC     HL
381A CD 60 3D      1169           CALL    calc_pointer_line_x_add
381D C3 C2 37      1170           JP      x_lp0
3820               1171   ;
3820 CD D6 3F      1172   x_lp8:  CALL    clr_botom
3823 CD 1A 3F      1173           CALL    view_free_colum
3826 AF            1174           XOR     A
3827 3D            1175           DEC     A
3828 32 E1 45      1176           LD      (flag_replace_all),A
382B C3 00 38      1177           JP      x_lp5
382E               1178   ;
382E DD 36 01 00   1179   x_lp9:  LD      (IX+1),0
3832 CD 9F 3E      1180           CALL    view_text
3835 C3 CC 35      1181           JP      c_cmde
3838               1182
3838               1183
3838               1184   calc_flash_locate:
3838 CD 4E 39      1185           CALL    calc_view_line
383B CD 3E 3B      1186           CALL    get_line
383E CD AD 3A      1187           CALL    calc_cursor_x
3841 CD 13 39      1188           CALL    calc_view_offset
3844 DD 36 01 00   1189           LD      (IX+1),0
3848 CD 9F 3E      1190           CALL    view_text
384B CD 0F 3B      1191           CALL    calc_locate_x
384E CD 2C 3B      1192           CALL    calc_locate_y
3851 C9            1193           RET
3852               1194
3852               1195
3852               1196   ;-------------------------------------
3852               1197   ; TEXT OP
3852               1198
3852               1199
3852               1200   top_cursor:
3852 AF            1201           XOR     A
3853 32 AE 45      1202           LD      (pointer_x),A
3856 CD AD 3A      1203           CALL    calc_cursor_x
3859 C9            1204           RET
385A               1205
385A               1206
385A               1207   botom_cursor:
385A AF            1208           XOR     A
385B 3D            1209           DEC     A
385C 32 B3 45      1210           LD      (cursor_x),A
385F CD DC 3A      1211           CALL    calc_pointer_x
3862 C9            1212           RET
3863               1213
3863               1214
3863               1215   ; Find strings
3863               1216   ;
3863               1217   find_str:
3863 CD D4 38      1218           CALL    calc_pointer
3866 ED 5B D9 45   1219           LD      DE,(find_b_adr)
386A EB            1220           EX      DE,HL
386B CD 94 1F      1221           CALL    _PEEK
386E 23            1222           INC     HL
386F 47            1223           LD      B,A
3870 1A            1224   f_str1: LD      A,(DE)
3871 13            1225           INC     DE
3872 B7            1226           OR      A
3873 CA 46 31      1227           JP      Z,scf_ret
3876 B8            1228           CP      B
3877 20 F7         1229           JR      NZ,f_str1
3879 E5            1230           PUSH    HL
387A D5            1231           PUSH    DE
387B CD 94 1F      1232   f_str3: CALL    _PEEK
387E B7            1233           OR      A
387F 28 0B         1234           JR      Z,f_str2
3881 4F            1235           LD      C,A
3882 1A            1236           LD      A,(DE)
3883 13            1237           INC     DE
3884 23            1238           INC     HL
3885 B9            1239           CP      C
3886 28 F3         1240           JR      Z,f_str3
3888 D1            1241           POP     DE
3889 E1            1242           POP     HL
388A 18 E4         1243           JR      f_str1
388C E1            1244   f_str2: POP     HL
388D D1            1245           POP     DE
388E 2B            1246           DEC     HL
388F CD 60 3D      1247           CALL    calc_pointer_line_x_add
3892 B7            1248           OR      A
3893 C9            1249           RET
3894               1250
3894               1251
3894               1252   ; Replace strings
3894               1253   ;
3894               1254   replace:
3894 3A E3 45      1255           LD      A,(replace_str_len)
3897 47            1256           LD      B,A
3898 3A E2 45      1257           LD      A,(find_str_len)
389B 90            1258           SUB     B
389C 28 04         1259           JR      Z,rep0
389E 38 14         1260           JR      C,rep2
38A0 18 21         1261           JR      rep3
38A2               1262   ;
38A2 CD D4 38      1263   rep0:   CALL    calc_pointer
38A5 ED 5B DB 45   1264           LD      DE,(replace_b_adr)
38A9 EB            1265           EX      DE,HL
38AA CD 94 1F      1266   rep1:   CALL    _PEEK
38AD B7            1267           OR      A
38AE C8            1268           RET     Z
38AF 12            1269           LD      (DE),A
38B0 23            1270           INC     HL
38B1 13            1271           INC     DE
38B2 18 F6         1272           JR      rep1
38B4               1273   ;
38B4 ED 44         1274   rep2:   NEG
38B6 47            1275           LD      B,A
38B7 CD D4 38      1276           CALL    calc_pointer
38BA 16 00         1277           LD      D,0
38BC 58            1278           LD      E,B
38BD CD DA 3B      1279           CALL    move_text_plus
38C0 D8            1280           RET     C
38C1 18 DF         1281           JR      rep0
38C3               1282   ;
38C3 47            1283   rep3:   LD      B,A
38C4 CD D4 38      1284           CALL    calc_pointer
38C7 3A E2 45      1285           LD      A,(find_str_len)
38CA 16 00         1286           LD      D,0
38CC 5F            1287           LD      E,A
38CD 19            1288           ADD     HL,DE
38CE 58            1289           LD      E,B
38CF CD 0D 3C      1290           CALL    move_text_minus
38D2 18 CE         1291           JR      rep0
38D4               1292
38D4               1293
38D4               1294   ; Calcurate pointer
38D4               1295   ;
```

▶子供(5歳と7歳)にX68000 PROを奪われてしまいました。家にはPC-9801もあるのに「わかる子供にはよいものがわかる」のかな？　西尾　雅也(30)神奈川県

```
38D4                1296  calc_pointer:
38D4 2A B1 45       1297          LD      HL,(pointer_adr)
38D7 3A AE 45       1298          LD      A,(pointer_x)
38DA 16 00          1299          LD      D,0
38DC 5F             1300          LD      E,A
38DD 19             1301          ADD     HL,DE
38DE C9             1302          RET
38DF                1303
38DF                1304
38DF                1305  ; Clear text.
38DF                1306  ;
38DF                1307  text_clear:
38DF 2A 84 45       1308          LD      HL,(text_start)
38E2 AF             1309          XOR     A
38E3 32 90 45       1310          LD      (edited_file),A
38E6 77             1311          LD      (HL),A
38E7 CD EB 38       1312          CALL    calc_text_size
38EA C9             1313          RET
38EB                1314
38EB                1315
38EB                1316  ; Calcurate text size.
38EB                1317  ;
38EB                1318  calc_text_size:
38EB 2A 84 45       1319          LD      HL,(text_start)
38EE CD 71 3A       1320  ctsl:   CALL    next_line
38F1 30 FB          1321          JR      NC,ctsl
38F3 22 86 45       1322          LD      (text_end),HL
38F6 ED 5B 84 45    1323          LD      DE,(text_start)
38FA B7             1324          OR      A
38FB ED 52          1325          SBC     HL,DE
38FD 23             1326          INC     HL
38FE 22 8A 45       1327          LD      (text_size),HL
3901 CD 05 39       1328          CALL    calc_text_free
3904 C9             1329          RET
3905                1330
3905                1331
3905                1332  ; Calucurate free size.
3905                1333  ;
3905                1334  calc_text_free:
3905 2A 88 45       1335          LD      HL,(text_max)
3908 ED 5B 86 45    1336          LD      DE,(text_end)
390C B7             1337          OR      A
390D ED 52          1338          SBC     HL,DE
390F 22 8C 45       1339          LD      (text_free),HL
3912 C9             1340          RET
3913                1341
3913                1342
3913                1343  calc_view_offset:
3913 3A A9 45       1344          LD      A,(view_offset)
3916 67             1345          LD      H,A
3917 3A 93 45       1346          LD      A,(console_width)
391A 3D             1347          DEC     A
391B 6F             1348          LD      L,A
391C 3A 06 30       1349          LD      A,(view_offset_w)
391F 57             1350          LD      D,A
3920 3A B3 45       1351          LD      A,(cursor_x)
3923 5F             1352          LD      E,A
3924 94             1353          SUB     H
3925 38 05          1354          JR      C,cvo1
3927 95             1355          SUB     L
3928 30 18          1356          JR      NC,cvo3
392A B7             1357          OR      A
392B C9             1358          RET
392C                1359  ;
392C 7C             1360  cvo1:   LD      A,H
392D 92             1361          SUB     D
392E 38 0B          1362          JR      C,cvo2
3930 32 A9 45       1363          LD      (view_offset),A
3933 67             1364          LD      H,A
3934 7B             1365          LD      A,E
3935 94             1366          SUB     H
3936 38 F4          1367          JR      C,cvo1
3938 C3 46 31       1368          JP      scf_ret
393B AF             1369  cvo2:   XOR     A
393C 32 A9 45       1370          LD      (view_offset),A
393F C3 46 31       1371          JP      scf_ret
3942                1372  ;
3942 7C             1373  cvo3:   LD      A,H
3943 82             1374          ADD     A,D
3944 32 A9 45       1375          LD      (view_offset),A
3947 67             1376          LD      H,A
3948 7B             1377          LD      A,E
3949 94             1378          SUB     H
394A 95             1379          SUB     L
394B 30 F5          1380          JR      NC,cvo3
394D C9             1381          RET
394E                1382
394E                1383
394E                1384  ; Calcurate line number of start of text viewer.
394E                1385  ;
394E                1386  calc_view_line:
394E 2A AF 45       1387          LD      HL,(pointer_line)
3951 ED 5B AA 45    1388          LD      DE,(view_line)
3955 B7             1389          OR      A
3956 E5             1390          PUSH    HL
3957 ED 52          1391          SBC     HL,DE
3959 E1             1392          POP     HL
395A 38 2F          1393          JR      C,cvl1
395C 06 00          1394          LD      B,0
395E 3A 94 45       1395          LD      A,(console_length)
3961 4F             1396          LD      C,A
3962 0D             1397          DEC     C
3963 E5             1398          PUSH    HL
3964 EB             1399          EX      DE,HL
3965 09             1400          ADD     HL,BC
3966 B7             1401          OR      A
3967 ED 52          1402          SBC     HL,DE
3969 E1             1403          POP     HL
396A D2 48 31       1404          JP      NC,rcf_ret
396D                1405  ;
396D B7             1406          OR      A
396E ED 42          1407          SBC     HL,BC
3970 ED 5B AA 45    1408          LD      DE,(view_line)
3974 22 AA 45       1409          LD      (view_line),HL
3977 ED 52          1410          SBC     HL,DE
3979 44             1411          LD      B,H
397A 4D             1412          LD      C,L
397B 2A AC 45       1413          LD      HL,(view_adr)
397E CD 71 3A       1414  cvl3:   CALL    next_line
3981 0B             1415          DEC     BC
3982 78             1416          LD      A,B
3983 B1             1417          OR      C
3984 20 F8          1418          JR      NZ,cvl3
3986 22 AC 45       1419          LD      (view_adr),HL
3989 37             1420          SCF
398A C9             1421          RET
398B                1422  ;
398B 22 AA 45       1423  cvl1:   LD      (view_line),HL
398E 2A B1 45       1424          LD      HL,(pointer_adr)
3991 22 AC 45       1425          LD      (view_adr),HL
3994 37             1426          SCF
3995 C9             1427          RET
3996                1428
3996                1429
3996                1430  calc_pointer_address:
3996 ED 4B AF 45    1431          LD      BC,(pointer_line)
399A CD A5 39       1432          CALL    calc_address
399D 22 B1 45       1433          LD      (pointer_adr),HL
39A0 ED 53 AF 45    1434          LD      (pointer_line),DE
39A4 C9             1435          RET
39A5                1436
39A5                1437
39A5                1438  calc_address:
39A5 2A 84 45       1439          LD      HL,(text_start)
39A8 11 00 00       1440          LD      DE,0
39AB D5             1441  ctp2:   PUSH    DE
39AC EB             1442          EX      DE,HL
39AD B7             1443          OR      A
39AE ED 42          1444          SBC     HL,BC
39B0 7C             1445          LD      A,H
39B1 B5             1446          OR      L
39B2 EB             1447          EX      DE,HL
39B3 D1             1448          POP     DE
39B4 C8             1449          RET     Z
39B5 CD 71 3A       1450          CALL    next_line
39B8 DA 46 31       1451          JP      C,scf_ret
39BB 13             1452          INC     DE
39BC 18 ED          1453          JR      ctp2
39BE                1454
39BE                1455
39BE                1456  ; Move pointer to the right.
39BE                1457  ;
39BE                1458  pointer_x_inc:
39BE CD 78 3C       1459          CALL    calc_pointer_inline
39C1 7E             1460          LD      A,(HL)
39C2 FE 0D          1461          CP      __CR
39C4 CA 46 31       1462          JP      Z,scf_ret
39C7 B7             1463          OR      A
39C8 CA 46 31       1464          JP      Z,scf_ret
39CB 79             1465          LD      A,C
39CC 3C             1466          INC     A
39CD 32 AE 45       1467          LD      (pointer_x),A
39D0 C9             1468          RET
39D1                1469
39D1                1470
39D1                1471  ; Move pointer to the left.
39D1                1472  ;
39D1                1473  pointer_x_dec:
39D1 3A AE 45       1474          LD      A,(pointer_x)
39D4 B7             1475          OR      A
39D5 CA 46 31       1476          JP      Z,scf_ret
39D8 3D             1477          DEC     A
39D9 32 AE 45       1478          LD      (pointer_x),A
39DC C9             1479          RET
39DD                1480
39DD                1481
39DD                1482  ; Move pointer to the next word.
39DD                1483  ;
39DD                1484  word_skip_foward:
39DD CD 78 3C       1485          CALL    calc_pointer_inline
39E0 7E             1486          LD      A,(HL)
39E1 FE 0D          1487          CP      __CR
39E3 CA 46 31       1488          JP      Z,scf_ret
39E6 7E             1489  wsf1:   LD      A,(HL)
39E7 23             1490          INC     HL
39E8 FE 0D          1491          CP      __CR
39EA 28 1B          1492          JR      Z,wsf3
39EC B7             1493          OR      A
39ED 28 18          1494          JR      Z,wsf3
39EF 0C             1495          INC     C
39F0 CD 35 3A       1496          CALL    skip_character
39F3 20 F1          1497          JR      NZ,wsf1
39F5 7E             1498  wsf2:   LD      A,(HL)
39F6 23             1499          INC     HL
39F7 FE 0D          1500          CP      __CR
39F9 CA 07 3A       1501          JP      Z,wsf3
39FC B7             1502          OR      A
39FD CA 07 3A       1503          JP      Z,wsf3
3A00 0C             1504          INC     C
3A01 CD 35 3A       1505          CALL    skip_character
3A04 28 EF          1506          JR      Z,wsf2
3A06 0D             1507          DEC     C
3A07 79             1508  wsf3:   LD      A,C
3A08 32 AE 45       1509          LD      (pointer_x),A
3A0B B7             1510          OR      A
3A0C C9             1511          RET
3A0D                1512
3A0D                1513
3A0D                1514  ; Move pointer to the back word.
3A0D                1515  ;
3A0D                1516  word_skip_back:
3A0D CD 78 3C       1517          CALL    calc_pointer_inline
3A10 79             1518          LD      A,C
3A11 B7             1519          OR      A
3A12 CA 46 31       1520          JP      Z,scf_ret
3A15 2B             1521  wsb1:   DEC     HL
3A16 0D             1522          DEC     C
3A17 79             1523          LD      A,C
3A18 B7             1524          OR      A
3A19 28 06          1525          JR      Z,wsb2
3A1B 7E             1526          LD      A,(HL)
3A1C CD 35 3A       1527          CALL    skip_character
3A1F 28 F4          1528          JR      Z,wsb1
3A21 2B             1529  wsb2:   DEC     HL
3A22 0D             1530          DEC     C
3A23 7E             1531          LD      A,(HL)
3A24 CD 35 3A       1532          CALL    skip_character
3A27 28 06          1533          JR      Z,wsb5
3A29 79             1534          LD      A,C
3A2A B7             1535          OR      A
3A2B 28 03          1536          JR      Z,wsb3
3A2D 18 F2          1537          JR      wsb2
3A2F 0C             1538  wsb5:   INC     C
3A30 79             1539  wsb3:   LD      A,C
3A31 32 AE 45       1540          LD      (pointer_x),A
3A34 C9             1541          RET
3A35                1542
3A35                1543
3A35                1544  ; Check if skip letter.
3A35                1545  ;
3A35                1546  skip_character:
3A35 C5             1547          PUSH    BC
```


▶長年乗っていたバイクと自転車を新調することにしました。ちなみにバイクは6年，自転車にいたっては12年乗っていました。今年はすでに金がない状態，愛機X68000 XVIになにもしてやれなくなってしまいました。来年まで辛抱な。　　久米　豊信(26)群馬県

```
3A36 E5            1548            PUSH    HL
3A37 F5            1549            PUSH    AF
3A38 21 5D 43      1550            LD      HL,skip_character_table
3A3B 06 00         1551            LD      B,0
3A3D 3A 5C 43      1552            LD      A,(skip_character_number)
3A40 4F            1553            LD      C,A
3A41 F1            1554            POP     AF
3A42 ED B1         1555            CPIR
3A44 E1            1556            POP     HL
3A45 C1            1557            POP     BC
3A46 C9            1558            RET
3A47               1559
3A47               1560
3A47               1561    ; Move pointer to the next line.
3A47               1562    ;
3A47               1563    pointer_line_inc:
3A47 2A B1 45      1564            LD      HL,(pointer_adr)
3A4A CD 71 3A      1565            CALL    next_line
3A4D D8            1566            RET     C
3A4E 22 B1 45      1567            LD      (pointer_adr),HL
3A51 2A AF 45      1568            LD      HL,(pointer_line)
3A54 23            1569            INC     HL
3A55 22 AF 45      1570            LD      (pointer_line),HL
3A58 B7            1571            OR      A
3A59 C9            1572            RET
3A5A               1573
3A5A               1574
3A5A               1575    ; Move pointer to the back line.
3A5A               1576    ;
3A5A               1577    pointer_line_dec:
3A5A 2A AF 45      1578            LD      HL,(pointer_line)
3A5D 7C            1579            LD      A,H
3A5E B5            1580            OR      L
3A5F CA 46 31      1581            JP      Z,scf_ret
3A62 2B            1582            DEC     HL
3A63 22 AF 45      1583            LD      (pointer_line),HL
3A66 2A B1 45      1584            LD      HL,(pointer_adr)
3A69 CD 7D 3A      1585            CALL    back_line
3A6C D8            1586            RET     C
3A6D 22 B1 45      1587            LD      (pointer_adr),HL
3A70 C9            1588            RET
3A71               1589
3A71               1590
3A71               1591    next_line:
3A71 7E            1592            LD      A,(HL)
3A72 B7            1593            OR      A
3A73 CA 46 31      1594            JP      Z,scf_ret
3A76 23            1595            INC     HL
3A77 FE 0D         1596            CP      __CR
3A79 C8            1597            RET     Z
3A7A C3 71 3A      1598            JP      next_line
3A7D               1599
3A7D               1600
3A7D               1601    back_line:
3A7D ED 5B 84 45   1602            LD      DE,(text_start)
3A81 E5            1603            PUSH    HL
3A82 B7            1604            OR      A
3A83 ED 52         1605            SBC     HL,DE
3A85 7C            1606            LD      A,H
3A86 B5            1607            OR      L
3A87 E1            1608            POP     HL
3A88 CA 46 31      1609            JP      Z,scf_ret
3A8B 1B            1610            DEC     DE
3A8C 2B            1611            DEC     HL
3A8D E5            1612    bk_l1:  PUSH    HL
3A8E B7            1613            OR      A
3A8F ED 52         1614            SBC     HL,DE
3A91 7C            1615            LD      A,H
3A92 B5            1616            OR      L
3A93 E1            1617            POP     HL
3A94 CA 9D 3A      1618            JP      Z,bk_l2
3A97 2B            1619            DEC     HL
3A98 7E            1620            LD      A,(HL)
3A99 FE 0D         1621            CP      __CR
3A9B 20 F0         1622            JR      NZ,bk_l1
3A9D 23            1623    bk_l2:  INC     HL
3A9E C9            1624            RET
3A9F               1625
3A9F               1626
3A9F               1627    save_cursor_x:
3A9F 3A B3 45      1628            LD      A,(cursor_x)
3AA2 32 B4 45      1629            LD      (cursor_x_keep),A
3AA5 C9            1630            RET
3AA6               1631
3AA6               1632
3AA6               1633    load_cursor_x:
3AA6 3A B4 45      1634            LD      A,(cursor_x_keep)
3AA9 32 B3 45      1635            LD      (cursor_x),A
3AAC C9            1636            RET
3AAD               1637
3AAD               1638
3AAD               1639    ; Calcurate colum by pointer.
3AAD               1640    ;
3AAD               1641    calc_cursor_x:
3AAD 21 FC 46      1642            LD      HL,line_buffer
3AB0 3A AE 45      1643            LD      A,(pointer_x)
3AB3 4F            1644            LD      C,A
3AB4 06 00         1645            LD      B,0
3AB6 11 FC 45      1646            LD      DE,tab_info_table
3AB9               1647    ;
3AB9 79            1648    ccx1:   LD      A,C
3ABA B7            1649            OR      A
3ABB 28 1A         1650            JR      Z,ccx3
3ABD 7E            1651            LD      A,(HL)
3ABE FE 09         1652            CP      __TAB
3AC0 28 0D         1653            JR      Z,ccx2
3AC2 FE 0D         1654            CP      __CR
3AC4 28 11         1655            JR      Z,ccx3
3AC6 B7            1656            OR      A
3AC7 28 0E         1657            JR      Z,ccx3
3AC9 04            1658            INC     B
3ACA 13            1659            INC     DE
3ACB 0D            1660    ccx4:   DEC     C
3ACC 23            1661            INC     HL
3ACD 18 EA         1662            JR      ccx1
3ACF               1663    ;
3ACF 13            1664    ccx2:   INC     DE
3AD0 04            1665            INC     B
3AD1 1A            1666            LD      A,(DE)
3AD2 B7            1667            OR      A
3AD3 28 FA         1668            JR      Z,ccx2
3AD5 18 F4         1669            JR      ccx4
3AD7               1670    ;
3AD7 78            1671    ccx3:   LD      A,B
3AD8 32 B3 45      1672            LD      (cursor_x),A
3ADB C9            1673            RET
3ADC               1674
3ADC               1675
3ADC               1676    ; Calculate position of pointer on line buffer.
3ADC               1677    ;
3ADC               1678    calc_pointer_x:
3ADC 21 FC 46      1679            LD      HL,line_buffer
3ADF 3A B3 45      1680            LD      A,(cursor_x)
3AE2 4F            1681            LD      C,A
3AE3 06 00         1682            LD      B,0
3AE5 11 FC 45      1683            LD      DE,tab_info_table
3AE8               1684    ;
3AE8 79            1685    cpx1:   LD      A,C
3AE9 B7            1686            OR      A
3AEA 28 1E         1687            JR      Z,cpx3
3AEC 7E            1688            LD      A,(HL)
3AED FE 09         1689            CP      __TAB
3AEF 28 0D         1690            JR      Z,cpx2
3AF1 FE 0D         1691            CP      __CR
3AF3 28 15         1692            JR      Z,cpx3
3AF5 B7            1693            OR      A
3AF6 28 12         1694            JR      Z,cpx3
3AF8 13            1695            INC     DE
3AF9 0D            1696            DEC     C
3AFA 04            1697    cpx4:   INC     B
3AFB 23            1698            INC     HL
3AFC 18 EA         1699            JR      cpx1
3AFE               1700    ;
3AFE 0D            1701    cpx2:   DEC     C
3AFF 13            1702            INC     DE
3B00 1A            1703            LD      A,(DE)
3B01 B7            1704            OR      A
3B02 20 F6         1705            JR      NZ,cpx4
3B04 79            1706            LD      A,C
3B05 B7            1707            OR      A
3B06 28 02         1708            JR      Z,cpx3
3B08 18 F4         1709            JR      cpx2
3B0A               1710    ;
3B0A 78            1711    cpx3:   LD      A,B
3B0B 32 AE 45      1712            LD      (pointer_x),A
3B0E C9            1713            RET
3B0F               1714
3B0F               1715
3B0F               1716    ; Calcutate cursor position: x on screen.
3B0F               1717    ;
3B0F               1718    calc_locate_x:
3B0F 3A A9 45      1719            LD      A,(view_offset)
3B12 6F            1720            LD      L,A
3B13 3A 91 45      1721            LD      A,(console_x)
3B16 67            1722            LD      H,A
3B17 3A 93 45      1723            LD      A,(console_width)
3B1A 3D            1724            DEC     A
3B1B 57            1725            LD      D,A
3B1C 3A B3 45      1726            LD      A,(cursor_x)
3B1F 95            1727            SUB     L
3B20 5F            1728            LD      E,A
3B21 D8            1729            RET     C
3B22 92            1730            SUB     D
3B23 D2 46 31      1731            JP      NC,scf_ret
3B26 7B            1732            LD      A,E
3B27 84            1733            ADD     A,H
3B28 32 B5 45      1734            LD      (locate_xy),A
3B2B C9            1735            RET
3B2C               1736
3B2C               1737
3B2C               1738    ; Calcurate cursor position: y on scrren.
3B2C               1739    ;
3B2C               1740    calc_locate_y:
3B2C 2A AF 45      1741            LD      HL,(pointer_line)
3B2F ED 4B AA 45   1742            LD      BC,(view_line)
3B33 B7            1743            OR      A
3B34 ED 42         1744            SBC     HL,BC
3B36 3A 92 45      1745            LD      A,(console_y)
3B39 85            1746            ADD     A,L
3B3A 32 B6 45      1747            LD      (locate_xy+1),A
3B3D C9            1748            RET
3B3E               1749
3B3E               1750
3B3E               1751    ; Get line from text to line buffer.
3B3E               1752    ;
3B3E               1753    get_line:
3B3E CD 4A 3B      1754            CALL    get_l0
3B41 79            1755            LD      A,C
3B42 32 CE 45      1756            LD      (line_length),A
3B45 AF            1757            XOR     A
3B46 32 CF 45      1758            LD      (flag_edited),A
3B49 C9            1759            RET
3B4A               1760    ;
3B4A 2A B1 45      1761    get_l0: LD      HL,(pointer_adr)
3B4D 11 FC 46      1762            LD      DE,line_buffer
3B50 FD 21 FC 45   1763            LD      IY,tab_info_table
3B54 06 01         1764            LD      B,1
3B56 0E 01         1765            LD      C,1
3B58 7E            1766    get_l1: LD      A,(HL)
3B59 12            1767            LD      (DE),A
3B5A FE 0D         1768            CP      __CR
3B5C C8            1769            RET     Z
3B5D B7            1770            OR      A
3B5E C8            1771            RET     Z
3B5F FE 09         1772            CP      __TAB
3B61 28 20         1773            JR      Z,get_l3
3B63 04            1774            INC     B
3B64 28 07         1775            JR      Z,get_l2
3B66 FD 23         1776            INC     IY
3B68 23            1777    get_l4: INC     HL
3B69 13            1778            INC     DE
3B6A 0C            1779            INC     C
3B6B 18 EB         1780            JR      get_l1
3B6D               1781    ;
3B6D 3E 0D         1782    get_l2: LD      A,__CR
3B6F 12            1783            LD      (DE),A
3B70 11 01 00      1784            LD      DE,1
3B73 E5            1785            PUSH    HL
3B74 CD DA 3B      1786            CALL    move_text_plus
3B77 E1            1787            POP     HL
3B78 D8            1788            RET     C
3B79 36 0D         1789            LD      (HL),__CR
3B7B DD 36 01 00   1790            LD      (IX+1),0
3B7F CD 9F 3E      1791            CALL    view_text
3B82 C9            1792            RET
3B83               1793    ;
3B83 04            1794    get_l3: INC     B
3B84 CA 6D 3B      1795            JP      Z,get_l2
3B87 FD 23         1796            INC     IY
3B89 FD 7E 00      1797            LD      A,(IY)
3B8C B7            1798            OR      A
3B8D 28 F4         1799            JR      Z,get_l3
```

▶以前のようなプログラミング重視の内容希望。　　三田　完(27)埼玉県

```
3B8F 18 D7          1800            JR      get_14
3B91                1801
3B91                1802
3B91                1803    ; Put line from line buffer to text.
3B91                1804    ;
3B91                1805    put_line:
3B91 3A CF 45       1806            LD      A,(flag_edited)
3B94 B7             1807            OR      A
3B95 C8             1808            RET     Z
3B96 CD 2C 3C       1809            CALL    calc_line_length
3B99 3A CE 45       1810            LD      A,(line_length)
3B9C 90             1811            SUB     B
3B9D 28 04          1812            JR      Z,put_10
3B9F 38 15          1813            JR      C,put_12
3BA1 18 24          1814            JR      put_13
3BA3                1815    ;
3BA3 2A B1 45       1816    put_10: LD      HL,(pointer_adr)
3BA6 11 FC 46       1817            LD      DE,line_buffer
3BA9 1A             1818    put_11: LD      A,(DE)
3BAA 77             1819            LD      (HL),A
3BAB FE 0D          1820            CP      __CR
3BAD 28 26          1821            JR      Z,put_14
3BAF B7             1822            OR      A
3BB0 28 23          1823            JR      Z,put_14
3BB2 23             1824            INC     HL
3BB3 13             1825            INC     DE
3BB4 18 F3          1826            JR      put_11
3BB6                1827    ;
3BB6 ED 44          1828    put_12: NEG
3BB8 16 00          1829            LD      D,0
3BBA 5F             1830            LD      E,A
3BBB 2A B1 45       1831            LD      HL,(pointer_adr)
3BBE CD 71 3A       1832            CALL    next_line
3BC1 CD DA 3B       1833            CALL    move_text_plus
3BC4 D8             1834            RET     C
3BC5 18 DC          1835            JR      put_10
3BC7                1836    ;
3BC7 16 00          1837    put_13: LD      D,0
3BC9 5F             1838            LD      E,A
3BCA 2A B1 45       1839            LD      HL,(pointer_adr)
3BCD CD 71 3A       1840            CALL    next_line
3BD0 CD 0D 3C       1841            CALL    move_text_minus
3BD3 18 CE          1842            JR      put_10
3BD5                1843    ;
3BD5 CD 0E 3D       1844    put_14: CALL    edited_text
3BD8 B7             1845            OR      A
3BD9 C9             1846            RET
3BDA                1847
3BDA                1848
3BDA                1849    ; Text move form HL to HL+DE.
3BDA                1850    ;
3BDA                1851    move_text_plus:
3BDA E5             1852            PUSH    HL
3BDB 2A 8C 45       1853            LD      HL,(text_free)
3BDE B7             1854            OR      A
3BDF ED 52          1855            SBC     HL,DE
3BE1 E1             1856            POP     HL
3BE2 38 24          1857            JR      C,mtp1
3BE4 D5             1858            PUSH    DE
3BE5 ED 5B 86 45    1859            LD      DE,(text_end)
3BE9 EB             1860            EX      DE,HL
3BEA B7             1861            OR      A
3BEB ED 52          1862            SBC     HL,DE
3BED 44             1863            LD      B,H
3BEE 4D             1864            LD      C,L
3BEF 03             1865            INC     BC
3BF0 E1             1866            POP     HL
3BF1 ED 5B 86 45    1867            LD      DE,(text_end)
3BF5 19             1868            ADD     HL,DE
3BF6 EB             1869            EX      DE,HL
3BF7 ED 53 86 45    1870            LD      (text_end),DE
3BFB ED B8          1871            LDDR
3BFD CD 0E 3D       1872    mtp0:   CALL    edited_text
3C00 CD 05 39       1873            CALL    calc_text_free
3C03 CD 71 3F       1874            CALL    view_free
3C06 F7             1875            OR      A
3C07 C9             1876            RET
3C08 CD C6 3F       1877    mtp1:   CALL    warn_mem_over
3C0B 37             1878            SCF
3C0C C9             1879            RET
3C0D                1880
3C0D                1881
3C0D                1882    ; Text move from HL to HL-DE.
3C0D                1883    ;
3C0D                1884    move_text_minus:
3C0D E5             1885            PUSH    HL
3C0E D5             1886            PUSH    DE
3C0F 54             1887            LD      D,H
3C10 5D             1888            LD      E,L
3C11 2A 86 45       1889            LD      HL,(text_end)
3C14 B7             1890            OR      A
3C15 ED 52          1891            SBC     HL,DE
3C17 44             1892            LD      B,H
3C18 4D             1893            LD      C,L
3C19 03             1894            INC     BC
3C1A D1             1895            POP     DE
3C1B E1             1896            POP     HL
3C1C E5             1897            PUSH    HL
3C1D ED 52          1898            SBC     HL,DE
3C1F 54             1899            LD      D,H
3C20 5D             1900            LD      E,L
3C21 E1             1901            POP     HL
3C22 ED B0          1902            LDIR
3C24 1B             1903            DEC     DE
3C25 ED 53 86 45    1904            LD      (text_end),DE
3C29 C3 FD 3B       1905            JP      mtp0
3C2C                1906
3C2C                1907
3C2C                1908    ; Calcurate line length on line buffer.
3C2C                1909    ;
3C2C                1910    calc_line_length:
3C2C 21 FC 46       1911            LD      HL,line_buffer
3C2F 06 00          1912            LD      B,0
3C31 04             1913    cll1:   INC     B
3C32 7E             1914            LD      A,(HL)
3C33 23             1915            INC     HL
3C34 FE 0D          1916            CP      __CR
3C36 28 05          1917            JR      Z,cll2
3C38 B7             1918            OR      A
3C39 28 02          1919            JR      Z,cll2
3C3B 18 F4          1920            JR      cll1
3C3D 78             1921    cll2:   LD      A,B
3C3E C9             1922            RET
3C3F                1923
3C3F                1924
3C3F                1925    ; Check if memory over: (text_end)+DE
3C3F                1926    ;
3C3F                1927    check_mem_over:
3C3F D5             1928            PUSH    DE
3C40 E5             1929            PUSH    HL
3C41 2A 86 45       1930            LD      HL,(text_end)
3C44 19             1931            ADD     HL,DE
3C45 ED 5B 88 45    1932            LD      DE,(text_max)
3C49 B7             1933            OR      A
3C4A ED 52          1934            SBC     HL,DE
3C4C E1             1935            POP     HL
3C4D D1             1936            POP     DE
3C4E D2 46 31       1937            JP      NC,scf_ret
3C51 B7             1938            OR      A
3C52 C9             1939            RET
3C53                1940
3C53                1941
3C53                1942    ; Check if colum is over than 255.
3C53                1943    ;
3C53                1944    line_over_check:
3C53 21 FC 46       1945            LD      HL,line_buffer
3C56 06 01          1946            LD      B,1
3C58 11 FC 45       1947            LD      DE,tab_info_table
3C5B                1948    ;
3C5B 7E             1949    loc1:   LD      A,(HL)
3C5C FE 09          1950            CP      __TAB
3C5E 28 0D          1951            JR      Z,loc2
3C60 FE 0D          1952            CP      __CR
3C62 C8             1953            RET     Z
3C63 B7             1954            OR      A
3C64 C8             1955            RET     Z
3C65 04             1956            INC     B
3C66 CA 46 31       1957            JP      Z,scf_ret
3C69 13             1958            INC     DE
3C6A 23             1959    loc4:   INC     HL
3C6B 18 EE          1960            JR      loc1
3C6D                1961    ;
3C6D 04             1962    loc2:   INC     B
3C6E CA 46 31       1963            JP      Z,scf_ret
3C71 13             1964            INC     DE
3C72 1A             1965            LD      A,(DE)
3C73 B7             1966            OR      A
3C74 28 F7          1967            JR      Z,loc2
3C76 18 F2          1968            JR      loc4
3C78                1969
3C78                1970
3C78                1971    ; Calcurate pointer on line buffer to HL.
3C78                1972    ;
3C78                1973    calc_pointer_inline:
3C78 21 FC 46       1974            LD      HL,line_buffer
3C7B 06 00          1975            LD      B,0
3C7D 3A AE 45       1976            LD      A,(pointer_x)
3C80 4F             1977            LD      C,A
3C81 09             1978            ADD     HL,BC
3C82 C9             1979            RET
3C83                1980
3C83                1981
3C83                1982    ; Exit out of escape mode.
3C83                1983    ;
3C83                1984    exit_escape_mode:
3C83 AF             1985            XOR     A
3C84 32 D6 45       1986            LD      (escape_mode),A
3C87 CD 8E 3C       1987            CALL    delete_mark
3C8A CD 4E 3F       1988            CALL    view_type_mode
3C8D C9             1989            RET
3C8E                1990
3C8E                1991
3C8E                1992    ; Delete mark
3C8E                1993    ;
3C8E                1994    delete_mark:
3C8E AF             1995            XOR     A
3C8F 32 BD 45       1996            LD      (flag_mark),A
3C92 CD 41 3F       1997            CALL    clr_mark_colum
3C95 C9             1998            RET
3C96                1999
3C96                2000
3C96                2001    ; Do mark.
3C96                2002    ;
3C96                2003    set_mark:
3C96 2A AF 45       2004            LD      HL,(pointer_line)
3C99 22 C0 45       2005            LD      (mark_1_line),HL
3C9C 3A AE 45       2006            LD      A,(pointer_x)
3C9F 32 BE 45       2007            LD      (mark_1_x),A
3CA2 2A B1 45       2008            LD      HL,(pointer_adr)
3CA5 22 C2 45       2009            LD      (mark_1_adr),HL
3CA8 3A B3 45       2010            LD      A,(cursor_x)
3CAB 32 BF 45       2011            LD      (mark_1_cr_x),A
3CAE AF             2012            XOR     A
3CAF 3D             2013            DEC     A
3CB0 32 BD 45       2014            LD      (flag_mark),A
3CB3 CD 34 3F       2015            CALL    view_mark_colum
3CB6 CD 9A 3F       2016            CALL    view_mark_x
3CB9 CD A8 3F       2017            CALL    view_mark_line
3CBC C9             2018            RET
3CBD                2019
3CBD                2020
3CBD                2021    set_mark_2:
3CBD 2A AF 45       2022            LD      HL,(pointer_line)
3CC0 22 C6 45       2023            LD      (mark_2_line),HL
3CC3 3A AE 45       2024            LD      A,(pointer_x)
3CC6 32 C4 45       2025            LD      (mark_2_x),A
3CC9 2A B1 45       2026            LD      HL,(pointer_adr)
3CCC 22 C8 45       2027            LD      (mark_2_adr),HL
3CCF 3A B3 45       2028            LD      A,(cursor_x)
3CD2 32 C5 45       2029            LD      (mark_2_cr_x),A
3CD5 C9             2030            RET
3CD6                2031
3CD6                2032
3CD6                2033    get_mark_1:
3CD6 2A C0 45       2034            LD      HL,(mark_1_line)
3CD9 22 AF 45       2035            LD      (pointer_line),HL
3CDC 3A BE 45       2036            LD      A,(mark_1_x)
3CDF 32 AE 45       2037            LD      (pointer_x),A
3CE2 2A C2 45       2038            LD      HL,(mark_1_adr)
3CE5 22 B1 45       2039            LD      (pointer_adr),HL
3CE8 3A BF 45       2040            LD      A,(mark_1_cr_x)
3CEB 32 B3 45       2041            LD      (cursor_x),A
3CEE C9             2042            RET
3CEF                2043
3CEF                2044
3CEF                2045    get_mark_2:
3CEF 2A C6 45       2046            LD      HL,(mark_2_line)
3CF2 22 AF 45       2047            LD      (pointer_line),HL
3CF5 3A C4 45       2048            LD      A,(mark_2_x)
3CF8 32 AE 45       2049            LD      (pointer_x),A
3CFB 2A C8 45       2050            LD      HL,(mark_2_adr)
3CFE 22 B1 45       2051            LD      (pointer_adr),HL
```


▶ 6月号は初心者の僕でも結構わかりやすかった。X68000を買ってから，ずっとOh!X買ってるけど初めてよくわかった。これからも初心者にやさしくお願いします。
浅利　宗弘(16)奈良県

```
3D01 3A C5 45         2052          LD      A,(mark_2_cr_x)
3D04 32 B3 45         2053          LD      (cursor_x),A
3D07 C9               2054          RET
3D08                  2055
3D08                  2056
3D08                  2057  ; This line has edited.
3D08                  2058  ;
3D08                  2059  edited_line:
3D08 AF               2060          XOR     A
3D09 3D               2061          DEC     A
3D0A 32 CF 45         2062          LD      (flag_edited),A
3D0D C9               2063          RET
3D0E                  2064
3D0E                  2065
3D0E                  2066  ; This text has edited.
3D0E                  2067  ;
3D0E                  2068  edited_text:
3D0E AF               2069          XOR     A
3D0F 3D               2070          DEC     A
3D10 32 90 45         2071          LD      (edited_file),A
3D13 C9               2072          RET
3D14                  2073
3D14                  2074
3D14                  2075  ; Insert letter in line.
3D14                  2076  ;
3D14                  2077  insert_type:
3D14 F5               2078          PUSH    AF
3D15 CD 99 3D         2079          CALL    save_line_1
3D18 21 FC 46         2080  it3:    LD      HL,line_buffer
3D1B 24               2081          INC     H
3D1C 54               2082          LD      D,H
3D1D 5D               2083          LD      E,L
3D1E 2B               2084          DEC     HL
3D1F 3A AE 45         2085          LD      A,(pointer_x)
3D22 E5               2086          PUSH    HL
3D23 21 01 01         2087          LD      HL,101H
3D26 06 00            2088          LD      B,0
3D28 4F               2089          LD      C,A
3D29 B7               2090          OR      A
3D2A ED 42            2091          SBC     HL,BC
3D2C 44               2092          LD      B,H
3D2D 4D               2093          LD      C,L
3D2E E1               2094          POP     HL
3D2F ED B8            2095          LDDR
3D31 CD 78 3C         2096          CALL    calc_pointer_inline
3D34 F1               2097  it2:    POP     AF
3D35 77               2098          LD      (HL),A
3D36 CD 53 3C         2099          CALL    line_over_check
3D39 38 09            2100          JR      C,it1
3D3B 0C               2101          INC     C
3D3C 79               2102          LD      A,C
3D3D 32 AE 45         2103          LD      (pointer_x),A
3D40 CD 08 3D         2104          CALL    edited_line
3D43 C9               2105          RET
3D44 CD A7 3D         2106  it1:    CALL    load_line_1
3D47 CD B6 3F         2107          CALL    warn_line_over
3D4A 37               2108          SCF
3D4B C9               2109          RET
3D4C                  2110
3D4C                  2111
3D4C                  2112  ; Change letter.
3D4C                  2113  ;
3D4C                  2114  over_type:
3D4C F5               2115          PUSH    AF
3D4D CD 99 3D         2116          CALL    save_line_1
3D50 CD 78 3C         2117          CALL    calc_pointer_inline
3D53 7E               2118          LD      A,(HL)
3D54 FE 0D            2119          CP      13
3D56 CA 18 3D         2120          JP      Z,it3
3D59 B7               2121          OR      A
3D5A CA 18 3D         2122          JP      Z,it3
3D5D C3 34 3D         2123          JP      it2
3D60                  2124
3D60                  2125
3D60                  2126  ; Calcurate pointer_line and pointer_x by HL
3D60                  2127  ;
3D60                  2128  calc_pointer_line_x_add:
3D60 ED 4B AF 45      2129          LD      BC,(pointer_line)
3D64 0B               2130          DEC     BC
3D65 54               2131          LD      D,H
3D66 5D               2132          LD      E,L
3D67 2A B1 45         2133          LD      HL,(pointer_adr)
3D6A E5               2134  cplx0:  PUSH    HL
3D6B B7               2135          OR      A
3D6C ED 52            2136          SBC     HL,DE
3D6E F5               2137          PUSH    AF
3D6F 7C               2138          LD      A,H
3D70 B5               2139          OR      L
3D71 28 21            2140          JR      Z,cplx2
3D73 F1               2141          POP     AF
3D74 E1               2142          POP     HL
3D75 30 08            2143          JR      NC,cplx1
3D77 CD 71 3A         2144          CALL    next_line
3D7A 03               2145          INC     BC
3D7B 38 02            2146          JR      C,cplx1
3D7D 18 EB            2147          JR      cplx0
3D7F D5               2148  cplx1:  PUSH    DE
3D80 CD 7D 3A         2149          CALL    back_line
3D83 D1               2150          POP     DE
3D84 ED 43 AF 45      2151  cplx3:  LD      (pointer_line),BC
3D88 22 B1 45         2152          LD      (pointer_adr),HL
3D8B EB               2153          EX      DE,HL
3D8C B7               2154          OR      A
3D8D ED 52            2155          SBC     HL,DE
3D8F 7D               2156          LD      A,L
3D90 32 AE 45         2157          LD      (pointer_x),A
3D93 C9               2158          RET
3D94 F1               2159  cplx2:  POP     AF
3D95 E1               2160          POP     HL
3D96 03               2161          INC     BC
3D97 18 EB            2162          JR      cplx3
3D99                  2163
3D99                  2164
3D99                  2165  ;-------------------------------------------
3D99                  2166  ; Undo/Copy buffer operation
3D99                  2167
3D99                  2168
3D99                  2169  save_line_1:
3D99 21 FC 46         2170          LD      HL,line_buffer
3D9C ED 5B D7 45      2171          LD      DE,(undo_1_b_adr)
3DA0 01 00 01         2172          LD      BC,100H
3DA3 CD 97 1F         2173          CALL    _POKE@
3DA6 C9               2174          RET
3DA7                  2175
3DA7                  2176
3DA7                  2177  load_line_1:
3DA7 21 FC 46         2178          LD      HL,line_buffer
3DAA ED 5B D7 45      2179          LD      DE,(undo_1_b_adr)
3DAE 01 00 01         2180          LD      BC,100H
3DB1 CD 91 1F         2181          CALL    _PEEK@
3DB4 C9               2182          RET
3DB5                  2183
3DB5                  2184
3DB5                  2185  ; Save to copy buffer from HL to HL+DE
3DB5                  2186  ;
3DB5                  2187  save_copy:
3DB5 E5               2188          PUSH    HL
3DB6 D5               2189          PUSH    DE
3DB7 CD EC 3D         2190          CALL    check_copy_size
3DBA 38 10            2191          JR      C,sc1
3DBC ED 53 CB 45      2192          LD      (copy_length),DE
3DC0 42               2193          LD      B,D
3DC1 4B               2194          LD      C,E
3DC2 ED 5B DD 45      2195          LD      DE,(copy_b_adr)
3DC6 CD 97 1F         2196          CALL    _POKE@
3DC9 D1               2197  sc0:    POP     DE
3DCA E1               2198          POP     HL
3DCB C9               2199          RET
3DCC CD C6 3F         2200  sc1:    CALL    warn_mem_over
3DCF 37               2201          SCF
3DD0 18 F7            2202          JR      sc0
3DD2                  2203
3DD2                  2204
3DD2                  2205  ; Load from copy buffer to HL
3DD2                  2206  ;
3DD2                  2207  load_copy:
3DD2 E5               2208          PUSH    HL
3DD3 D5               2209          PUSH    DE
3DD4 ED 5B CB 45      2210          LD      DE,(copy_length)
3DD8 E5               2211          PUSH    HL
3DD9 D5               2212          PUSH    DE
3DDA CD DA 3B         2213          CALL    move_text_plus
3DDD C1               2214          POP     BC
3DDE E1               2215          POP     HL
3DDF 38 08            2216          JR      C,l_c1
3DE1 ED 5B DD 45      2217          LD      DE,(copy_b_adr)
3DE5 CD 91 1F         2218          CALL    _PEEK@
3DE8 B7               2219          OR      A
3DE9 D1               2220  l_c1:   POP     DE
3DEA E1               2221          POP     HL
3DEB C9               2222          RET
3DEC                  2223
3DEC                  2224
3DEC                  2225  ; Check if over than buffer size
3DEC                  2226  ;
3DEC                  2227  check_copy_size:
3DEC E5               2228          PUSH    HL
3DED 2A DF 45         2229          LD      HL,(buffer_size)
3DF0 B7               2230          OR      A
3DF1 ED 52            2231          SBC     HL,DE
3DF3 E1               2232          POP     HL
3DF4 C9               2233          RET
3DF5                  2234
3DF5                  2235
3DF5                  2236  can_paste:
3DF5 AF               2237          XOR     A
3DF6 32 CD 45         2238          LD      (flag_cant_past),A
3DF9 C9               2239          RET
3DFA                  2240
3DFA                  2241
3DFA                  2242  cant_paste:
3DFA AF               2243          XOR     A
3DFB 3D               2244          DEC     A
3DFC 32 CD 45         2245          LD      (flag_cant_past),A
3DFF C9               2246          RET
3E00                  2247
3E00                  2248
3E00                  2249  swap_mark_pointer:
3E00 2A C6 45         2250          LD      HL,(mark_2_line)
3E03 ED 5B C0 45      2251          LD      DE,(mark_1_line)
3E07 B7               2252          OR      A
3E08 ED 52            2253          SBC     HL,DE
3E0A 28 0B            2254          JR      Z,smp1
3E0C D8               2255          RET     C
3E0D CD 44 3E         2256          CALL    swap_mark_pointer_line
3E10 CD 27 3E         2257          CALL    swap_mark_pointer_x
3E13 CD 61 3E         2258          CALL    save_pointer
3E16 C9               2259          RET
3E17 3A BE 45         2260  smp1:   LD      A,(mark_1_x)
3E1A 47               2261          LD      B,A
3E1B 3A C4 45         2262          LD      A,(mark_2_x)
3E1E B8               2263          CP      B
3E1F D8               2264          RET     C
3E20 CD 27 3E         2265          CALL    swap_mark_pointer_x
3E23 CD 61 3E         2266          CALL    save_pointer
3E26 C9               2267          RET
3E27                  2268
3E27                  2269
3E27                  2270  swap_mark_pointer_x:
3E27 3A BE 45         2271          LD      A,(mark_1_x)
3E2A 47               2272          LD      B,A
3E2B 3A C4 45         2273          LD      A,(mark_2_x)
3E2E 32 BE 45         2274          LD      (mark_1_x),A
3E31 78               2275          LD      A,B
3E32 32 C4 45         2276          LD      (mark_2_x),A
3E35 3A BF 45         2277          LD      A,(mark_1_cr_x)
3E38 47               2278          LD      B,A
3E39 3A C5 45         2279          LD      A,(mark_2_cr_x)
3E3C 32 BF 45         2280          LD      (mark_1_cr_x),A
3E3F 78               2281          LD      A,B
3E40 32 C5 45         2282          LD      (mark_2_cr_x),A
3E43 C9               2283          RET
3E44                  2284
3E44                  2285
3E44                  2286  swap_mark_pointer_line:
3E44 2A C0 45         2287          LD      HL,(mark_1_line)
3E47 ED 5B C6 45      2288          LD      DE,(mark_2_line)
3E4B ED 53 C0 45      2289          LD      (mark_1_line),DE
3E4F 22 C6 45         2290          LD      (mark_2_line),HL
3E52 2A C2 45         2291          LD      HL,(mark_1_adr)
3E55 ED 5B C8 45      2292          LD      DE,(mark_2_adr)
3E59 ED 53 C2 45      2293          LD      (mark_1_adr),DE
3E5D 22 C8 45         2294          LD      (mark_2_adr),HL
3E60 C9               2295          RET
3E61                  2296
3E61                  2297
3E61                  2298  save_pointer:
3E61 3A AE 45         2299          LD      A,(pointer_x)
3E64 32 B7 45         2300          LD      (u_pointer_x),A
3E67 3A B3 45         2301          LD      A,(cursor_x)
3E6A 32 BC 45         2302          LD      (u_cursor_x),A
3E6D 2A AF 45         2303          LD      HL,(pointer_line)
```

▶友人「パソコンなにもってんの～？」。私「え～68」。友人「えっ!? それってすごく古くない？」。私「そんなことない……(と思うよ)」。よくよく聞いてみると友人はNEC製だと思ったらしい。なんと答えれば勘違いされないんだろう。　飯田　雅代(18)東京都

```
3E70 22 B8 45       2304          LD      (u_pointer_line),HL
3E73 2A B1 45       2305          LD      HL,(pointer_adr)
3E76 22 BA 45       2306          LD      (u_pointer_adr),HL
3E79 C9             2307          RET
3E7A                2308
3E7A                2309
3E7A                2310 load_pointer:
3E7A 3A B7 45       2311          LD      A,(u_pointer_x)
3E7D 32 AE 45       2312          LD      (pointer_x),A
3E80 3A BC 45       2313          LD      A,(u_cursor_x)
3E83 32 B3 45       2314          LD      (cursor_x),A
3E86 2A B8 45       2315          LD      HL,(u_pointer_line)
3E89 22 AF 45       2316          LD      (pointer_line),HL
3E8C 2A BA 45       2317          LD      HL,(u_pointer_adr)
3E8F 22 B1 45       2318          LD      (pointer_adr),HL
3E92 CD 9F 3A       2319          CALL    save_cursor_x
3E95 C9             2320          RET
3E96                2321
3E96                2322
3E96                2323 ;-------------------------------------
3E96                2324 ; SCREEN OPE
3E96                2325
3E96                2326
3E96                2327 view_screen:
3E96 3E 0C          2328          LD      A,__CLS
3E98 CD F4 1F       2329          CALL    _PRINT
3E9B CD F3 3E       2330          CALL    view_colums
3E9E C9             2331          RET
3E9F                2332
3E9F                2333
3E9F                2334 view_text:
3E9F 2A AC 45       2335          LD      HL,(view_adr)
3EA2 3A 94 45       2336          LD      A,(console_length)
3EA5 4F             2337          LD      C,A
3EA6 AF             2338          XOR     A
3EA7 57             2339          LD      D,A
3EA8 47             2340          LD      B,A
3EA9 32 D1 45       2341          LD      (line_print_y),A
3EAC 22 D2 45       2342 vw_t1:   LD      (line_data_adr),HL
3EAF                2343 ;
3EAF 78             2344          LD      A,B
3EB0 32 D1 45       2345          LD      (line_print_y),A
3EB3 DD 72 00       2346          LD      (IX),D
3EB6 DD 7E 01       2347          LD      A,(IX+1)
3EB9 B7             2348          OR      A
3EBA 28 12          2349          JR      Z,vw_t4
3EBC C5             2350          PUSH    BC
3EBD E5             2351          PUSH    HL
3EBE ED 4B B1 45    2352          LD      BC,(pointer_adr)
3EC2 B7             2353          OR      A
3EC3 ED 42          2354          SBC     HL,BC
3EC5 7C             2355          LD      A,H
3EC6 B5             2356          OR      L
3EC7 E1             2357          POP     HL
3EC8 C1             2358          POP     BC
3EC9 C4 00 44       2359          CALL    NZ,line_print
3ECC 18 03          2360          JR      vw_t5
3ECE CD 00 44       2361 vw_t4:   CALL    line_print
3ED1 04             2362 vw_t5:   INC     B
3ED2 CD 71 3A       2363          CALL    next_line
3ED5 38 04          2364          JR      C,vw_t2
3ED7 0D             2365 vw_t3:   DEC     C
3ED8 20 D2          2366          JR      NZ,vw_t1
3EDA C9             2367          RET
3EDB                2368 ;
3EDB 16 02          2369 vw_t2:   LD      D,2
3EDD 18 F8          2370          JR      vw_t3
3EDF                2371
3EDF                2372
3EDF                2373 view_line_buffer:
3EDF 21 FC 46       2374          LD      HL,line_buffer
3EE2 22 D2 45       2375          LD      (line_data_adr),HL
3EE5 3A B6 45       2376          LD      A,(locate_xy+1)
3EE8 32 D1 45       2377          LD      (line_print_y),A
3EEB AF             2378          XOR     A
3EEC DD 77 00       2379          LD      (IX),A
3EEF CD 00 44       2380          CALL    line_print
3EF2 C9             2381          RET
3EF3                2382
3EF3                2383
3EF3                2384 view_colums:
3EF3 CD 03 3F       2385          CALL    view_filename
3EF6 CD 1A 3F       2386          CALL    view_free_colum
3EF9 CD 27 3F       2387          CALL    view_pointer_colum
3EFC CD 4E 3F       2388          CALL    view_type_mode
3EFF CD 71 3F       2389          CALL    view_free
3F02 C9             2390          RET
3F03                2391
3F03                2392
3F03                2393 view_filename:
3F03 2A 9B 45       2394          LD      HL,(v_fnam_xy)
3F06 CD 1E 20       2395          CALL    _LOC
3F09 3E 5B          2396          LD      A,"["
3F0B CD F4 1F       2397          CALL    _PRINT
3F0E 11 E8 45       2398          LD      DE,filename
3F11 CD E5 1F       2399          CALL    _MSX
3F14 3E 5D          2400          LD      A,"]"
3F16 CD F4 1F       2401          CALL    _PRINT
3F19 C9             2402          RET
3F1A                2403
3F1A                2404
3F1A                2405 view_free_colum:
3F1A 2A 95 45       2406          LD      HL,(v_free_xy)
3F1D CD 1E 20       2407          CALL    _LOC
3F20 11 D7 42       2408          LD      DE,free_colum
3F23 CD E5 1F       2409          CALL    _MSX
3F26 C9             2410          RET
3F27                2411
3F27                2412
3F27                2413 view_pointer_colum:
3F27 2A 97 45       2414          LD      HL,(v_pointer_xy)
3F2A CD 1E 20       2415          CALL    _LOC
3F2D 11 E4 42       2416          LD      DE,pointer_colum
3F30 CD E5 1F       2417          CALL    _MSX
3F33 C9             2418          RET
3F34                2419
3F34                2420
3F34                2421 view_mark_colum:
3F34 2A 99 45       2422          LD      HL,(v_mark_xy)
3F37 CD 1E 20       2423          CALL    _LOC
3F3A 11 F5 42       2424          LD      DE,mark_colum
3F3D CD E5 1F       2425          CALL    _MSX
3F40 C9             2426          RET
3F41                2427
3F41                2428
3F41                2429 clr_mark_colum:
```

```
3F41 2A 99 45       2430          LD      HL,(v_mark_xy)
3F44 CD 1E 20       2431          CALL    _LOC
3F47 11 06 43       2432          LD      DE,colum_clr
3F4A CD E5 1F       2433          CALL    _MSX
3F4D C9             2434          RET
3F4E                2435
3F4E                2436
3F4E                2437 view_type_mode:
3F4E 2A 9D 45       2438          LD      HL,(v_mode_xy)
3F51 CD 1E 20       2439          CALL    _LOC
3F54 3A D6 45       2440          LD      A,(escape_mode)
3F57 B7             2441          OR      A
3F58 28 07          2442          JR      Z,v_m1
3F5A 11 2D 43       2443          LD      DE,s_escape
3F5D CD E5 1F       2444 v_m0:    CALL    _MSX
3F60 C9             2445          RET
3F61 3A D5 45       2446 v_m1:    LD      A,(type_mode)
3F64 B7             2447          OR      A
3F65 28 05          2448          JR      Z,v_m2
3F67 11 17 43       2449          LD      DE,s_overtype
3F6A 18 F1          2450          JR      v_m0
3F6C 11 22 43       2451 v_m2:    LD      DE,s_insert
3F6F 18 EC          2452          JR      v_m0
3F71                2453
3F71                2454
3F71                2455 view_free:
3F71 2A 9F 45       2456          LD      HL,(free_xy)
3F74 CD 1E 20       2457          CALL    _LOC
3F77 2A 8C 45       2458          LD      HL,(text_free)
3F7A CD 5B 41       2459          CALL    print_HL
3F7D C9             2460          RET
3F7E                2461
3F7E                2462
3F7E                2463 view_cursor_x:
3F7E 2A A1 45       2464          LD      HL,(pointer_x_xy)
3F81 CD 1E 20       2465          CALL    _LOC
3F84 3A B3 45       2466          LD      A,(cursor_x)
3F87 3C             2467          INC     A
3F88 CD 9A 41       2468          CALL    print_A
3F8B C9             2469          RET
3F8C                2470
3F8C                2471
3F8C                2472 view_pointer_line:
3F8C 2A A3 45       2473          LD      HL,(pointer_line_xy)
3F8F CD 1E 20       2474          CALL    _LOC
3F92 2A AF 45       2475          LD      HL,(pointer_line)
3F95 23             2476          INC     HL
3F96 CD 5B 41       2477          CALL    print_HL
3F99 C9             2478          RET
3F9A                2479
3F9A                2480
3F9A                2481 view_mark_x:
3F9A 2A A5 45       2482          LD      HL,(mark_x_xy)
3F9D CD 1E 20       2483          CALL    _LOC
3FA0 3A BF 45       2484          LD      A,(mark_l_cr_x)
3FA3 3C             2485          INC     A
3FA4 CD 9A 41       2486          CALL    print_A
3FA7 C9             2487          RET
3FA8                2488
3FA8                2489
3FA8                2490 view_mark_line:
3FA8 2A A7 45       2491          LD      HL,(mark_line_xy)
3FAB CD 1E 20       2492          CALL    _LOC
3FAE 2A C0 45       2493          LD      HL,(mark_l_line)
3FB1 23             2494          INC     HL
3FB2 CD 5B 41       2495          CALL    print_HL
3FB5 C9             2496          RET
3FB6                2497
3FB6                2498
3FB6                2499 warn_line_over:
3FB6 2A 9D 45       2500          LD      HL,(v_mode_xy)
3FB9 CD 1E 20       2501          CALL    _LOC
3FBC 11 38 43       2502          LD      DE,s_line_over
3FBF CD E5 1F       2503          CALL    _MSX
3FC2 CD C4 1F       2504          CALL    _BELL
3FC5 C9             2505          RET
3FC6                2506
3FC6                2507
3FC6                2508 warn_mem_over:
3FC6 2A 9D 45       2509          LD      HL,(v_mode_xy)
3FC9 CD 1E 20       2510          CALL    _LOC
3FCC 11 43 43       2511          LD      DE,s_mem_over
3FCF CD E5 1F       2512          CALL    _MSX
3FD2 CD C4 1F       2513          CALL    _BELL
3FD5 C9             2514          RET
3FD6                2515
3FD6                2516
3FD6                2517 clr_botom:
3FD6 2A 9D 45       2518          LD      HL,(v_mode_xy)
3FD9 CD 1E 20       2519          CALL    _LOC
3FDC 3A 93 45       2520          LD      A,(console_width)
3FDF 3D             2521          DEC     A
3FE0 47             2522          LD      B,A
3FE1 CD F1 1F       2523 cb1:     CALL    _PRNTS
3FE4 10 FB          2524          DJNZ    cb1
3FE6 C9             2525          RET
3FE7                2526
3FE7                2527
3FE7                2528 view_botom:
3FE7 CD D6 3F       2529          CALL    clr_botom
3FEA CD 4E 3F       2530          CALL    view_type_mode
3FED CD 1A 3F       2531          CALL    view_free_colum
3FF0 CD 27 3F       2532          CALL    view_pointer_colum
3FF3 CD 8C 3F       2533          CALL    view_pointer_line
3FF6 CD 71 3F       2534          CALL    view_free
3FF9 C9             2535          RET
3FFA                2536
3FFA                2537
3FFA                2538 mes_file_open:
3FFA 11 35 42       2539          LD      DE,mes_2
3FFD CD E5 1F       2540          CALL    _MSX
4000 11 E8 45       2541          LD      DE,filename
4003 CD E5 1F       2542          CALL    _MSX
4006 CD EE 1F       2543          CALL    _LTNL
4009 C9             2544          RET
400A                2545
400A                2546
400A                2547 mes_file_close:
400A 11 42 42       2548          LD      DE,mes_3
400D CD E5 1F       2549          CALL    _MSX
4010 11 E8 45       2550          LD      DE,filename
4013 CD E5 1F       2551          CALL    _MSX
4016 CD EE 1F       2552          CALL    _LTNL
4019 C9             2553          RET
401A                2554
401A                2555 ;-------------------------------------
```

▶５月で20歳になってしまいました。考えてみれば，最初にOh!Xに載ったのが15歳。直後にいまのマシンを購入し，いまでも活用している。本当にすごいことです。

天達　雄一(20)広島県

```
401A                2556 ; Asks
401A                2557
401A                2558
401A                2559 ask_save:
401A CD D6 3F       2560         CALL    clr_botom
401D 2A 9D 45       2561         LD      HL,(v_mode_xy)
4020 CD 1E 20       2562         CALL    _LOC
4023 11 4E 42       2563         LD      DE,mes_4
4026 CD E5 1F       2564         CALL    _MSX
4029 CD 21 20       2565         CALL    _FLGET
402C F5             2566         PUSH    AF
402D CD E7 3F       2567         CALL    view_botom
4030 F1             2568         POP     AF
4031 E6 DF          2569         AND     0DFH
4033 FE 43          2570         CP      "C"
4035 CA 46 31       2571         JP      Z,scf_ret
4038 FE 4E          2572         CP      "N"
403A 28 06          2573         JR      Z,a_s2
403C FE 53          2574         CP      "S"
403E 28 06          2575         JR      Z,a_s3
4040 18 D8          2576         JR      ask_save
4042                2577 ;
4042 AF             2578 a_s2:   XOR     A
4043 3D             2579         DEC     A
4044 B7             2580         OR      A
4045 C9             2581         RET
4046                2582 ;
4046 AF             2583 a_s3:   XOR     A
4047 C9             2584         RET
4048                2585
4048                2586
4048                2587 pause:
4048 11 70 42       2588         LD      DE,mes_5
404B CD E5 1F       2589         CALL    _MSX
404E CD 21 20       2590         CALL    _FLGET
4051 C9             2591         RET
4052                2592
4052                2593
4052                2594 ;--------------------------------------
4052                2595 ; FILE OPE
4052                2596
4052                2597
4052                2598 new_file:
4052 3E 04          2599         LD      A,4             ; entry : DE
4054 CD A3 1F       2600 n_fl:   CALL    _FILE
4057 CD BC 40       2601         CALL    get_filename
405A CD DF 38       2602         CALL    text_clear
405D C9             2603         RET
405E                2604
405E                2605
405E                2606 file_open_r:
405E 3E 04          2607         LD      A,4             ; entry : DE
4060 CD A3 1F       2608         CALL    _FILE
4063 CD BC 40       2609         CALL    get_filename
4066 CD FA 3F       2610         CALL    mes_file_open
4069 CD 09 20       2611         CALL    _ROPEN
406C C9             2612         RET
406D                2613
406D                2614
406D                2615 file_open_w:
406D 3E 04          2616         LD      A,4             ; entry : DE
406F CD A3 1F       2617         CALL    _FILE
4072 CD BC 40       2618         CALL    get_filename
4075 CD 0A 40       2619         CALL    mes_file_close
4078 21 00 00       2620         LD      HL,0
407B 22 70 1F       2621         LD      (_DTADR),HL
407E 22 6E 1F       2622         LD      (_EXADR),HL
4081 CD EB 38       2623         CALL    calc_text_size
4084 2A 8A 45       2624         LD      HL,(text_size)
4087 22 72 1F       2625         LD      (_SIZE),HL
408A CD AF 1F       2626         CALL    _WOPEN
408D C9             2627         RET
408E                2628
408E                2629
408E                2630 file_read:
408E ED 5B 72 1F    2631         LD      DE,(_SIZE)
4092 2A 8E 45       2632         LD      HL,(text_area)
4095 ED 52          2633         SBC     HL,DE
4097 20 03          2634         JR      NZ,f_rl
4099 3E 0F          2635         LD      A,15
409B C9             2636         RET
409C 2A 84 45       2637 f_rl:   LD      HL,(text_start)
409F 22 70 1F       2638         LD      (_DTADR),HL
40A2 CD A6 1F       2639         CALL    _RDD
40A5 CD EB 38       2640         CALL    calc_text_size
40A8 C9             2641         RET
40A9                2642
40A9                2643
40A9                2644 file_write:
40A9 2A 84 45       2645         LD      HL,(text_start)
40AC 22 70 1F       2646         LD      (_DTADR),HL
40AF 22 6E 1F       2647         LD      (_EXADR),HL
40B2 2A 8A 45       2648         LD      HL,(text_size)
40B5 22 72 1F       2649         LD      (_SIZE),HL
40B8 CD AC 1F       2650         CALL    _WRD
40BB C9             2651         RET
40BC                2652
40BC                2653
40BC                2654 get_filename:
40BC 21 E8 45       2655         LD      HL,filename
40BF 3A 5D 1F       2656         LD      A,(_DSK)
40C2 77             2657         LD      (HL),A
40C3 23             2658         INC     HL
40C4 36 3A          2659         LD      (HL),":"
40C6 23             2660         INC     HL
40C7 ED 5B 74 1F    2661         LD      DE,(_IBFAD)
40CB 06 0D          2662         LD      B,__CR
40CD CD DB 40       2663         CALL    g_fl
40D0 36 2E          2664         LD      (HL),"."
40D2 23             2665         INC     HL
40D3 06 03          2666         LD      B,3
40D5 CD DB 40       2667         CALL    g_fl
40D8 36 00          2668         LD      (HL),0
40DA C9             2669         RET
40DB 13             2670 g_fl:   INC     DE
40DC 1A             2671         LD      A,(DE)
40DD 77             2672         LD      (HL),A
40DE 23             2673         INC     HL
40DF 05             2674         DEC     B
40E0 20 F9          2675         JR      NZ,g_fl
40E2 C9             2676         RET
40E3                2677
40E3                2678
40E3                2679 save_file:
40E3 CD 6D 40       2680         CALL    file_open_w
40E6 DA 1D 41       2681         JP      C,f_err
40E9 CD A9 40       2682         CALL    file_write
40EC DA 1D 41       2683         JP      C,f_err
40EF AF             2684         XOR     A
40F0 32 90 45       2685         LD      (edited_file),A
40F3 C9             2686         RET
40F4                2687
40F4                2688
40F4                2689 close_file:
40F4 3A 90 45       2690         LD      A,(edited_file)
40F7 B7             2691         OR      A
40F8 C8             2692         RET     Z
40F9 D5             2693         PUSH    DE
40FA CD 1A 40       2694         CALL    ask_save
40FD D1             2695         POP     DE
40FE D8             2696         RET     C
40FF C0             2697         RET     NZ
4100 3E 0C          2698         LD      A,__CLS
4102 CD F4 1F       2699         CALL    _PRINT
4105 11 E8 45       2700         LD      DE,filename
4108 CD E3 40       2701         CALL    save_file
410B C9             2702         RET
410C                2703
410C                2704
410C                2705 load_file:
410C CD 5E 40       2706         CALL    file_open_r
410F DA 1D 41       2707         JP      C,f_err
4112 CD 8E 40       2708         CALL    file_read
4115 DA 1D 41       2709         JP      C,f_err
4118 AF             2710         XOR     A
4119 32 90 45       2711         LD      (edited_file),A
411C C9             2712         RET
411D                2713
411D                2714
411D                2715 f_err:
411D FE 0F          2716         CP      15
411F 28 07          2717         JR      Z,f_err1
4121 CD 33 20       2718         CALL    _ERROR
4124 CD 48 40       2719 f_err0: CALL    pause
4127 C9             2720         RET
4128 11 7E 42       2721 f_err1: LD      DE,mes_6
412B CD E5 1F       2722         CALL    _MSX
412E CD C4 1F       2723         CALL    _BELL
4131 18 F1          2724         JR      f_err0
4133                2725
4133                2726
4133                2727 ;--------------------------------------
4133                2728 ; SUBROUTINES
4133                2729
4133                2730
4133                2731 division16:
4133 21 00 00       2732         LD      HL,0            ; DE <= DE / BC
4136 3E 10          2733         LD      A,16            ; HL <= DE MOD BC
4138 CB 23          2734 div161: SLA     E
413A CB 12          2735         RL      D
413C ED 6A          2736         ADC     HL,HL
413E ED 42          2737         SBC     HL,BC
4140 38 03          2738         JR      C,div162
4142 1C             2739         INC     E
4143 18 01          2740         JR      div163
4145 09             2741 div162: ADD     HL,BC
4146 3D             2742 div163: DEC     A
4147 20 EF          2743         JR      NZ,div161
4149 C9             2744         RET
414A                2745
414A                2746
414A                2747 division8:
414A 3E 00          2748         LD      A,0             ; D <= D / E
414C 06 08          2749         LD      B,8             ; A <= D MOD E
414E CB 22          2750 div81:  SLA     D
4150 8F             2751         ADC     A,A
4151 9B             2752         SBC     A,E
4152 38 03          2753         JR      C,div82
4154 14             2754         INC     D
4155 18 01          2755         JR      div83
4157 83             2756 div82:  ADD     A,E
4158 10 F4          2757 div83:  DJNZ    div81
415A C9             2758         RET
415B                2759
415B                2760
415B                2761 print_HL:
415B E5             2762         PUSH    HL
415C D5             2763         PUSH    DE
415D C5             2764         PUSH    BC
415E F5             2765         PUSH    AF
415F AF             2766         XOR     A
4160 DD 77 00       2767         LD      (IX),A
4163 01 10 27       2768         LD      BC,10000
4166 EB             2769         EX      DE,HL
4167 CD 33 41       2770         CALL    division16
416A 7B             2771         LD      A,E
416B CD BF 41       2772         CALL    print_number
416E 01 E8 03       2773         LD      BC,1000
4171 EB             2774         EX      DE,HL
4172 CD 33 41       2775         CALL    division16
4175 7B             2776         LD      A,E
4176 CD BF 41       2777         CALL    print_number
4179 01 64 00       2778         LD      BC,100
417C EB             2779         EX      DE,HL
417D CD 33 41       2780         CALL    division16
4180 7B             2781         LD      A,E
4181 CD BF 41       2782         CALL    print_number
4184 01 0A 00       2783         LD      BC,10
4187 EB             2784         EX      DE,HL
4188 CD 33 41       2785         CALL    division16
418B 7B             2786         LD      A,E
418C CD BF 41       2787         CALL    print_number
418F 7D             2788         LD      A,L
4190 C6 30          2789         ADD     A,30H
4192 CD F4 1F       2790         CALL    _PRINT
4195 F1             2791         POP     AF
4196 C1             2792         POP     BC
4197 D1             2793         POP     DE
4198 E1             2794         POP     HL
4199 C9             2795         RET
419A                2796
419A                2797
419A                2798 print_A:
419A D5             2799         PUSH    DE
419B F5             2800         PUSH    AF
419C 57             2801         LD      D,A
419D AF             2802         XOR     A
419E DD 77 00       2803         LD      (IX),A
41A1 1E 64          2804         LD      E,100
41A3 CD 4A 41       2805         CALL    division8
41A6 5F             2806         LD      E,A
41A7 7A             2807         LD      A,D
```

▶おお！ X68000 CompactXVIがオープン価格になってる。 宮津 隆幸(27)東京都

```
2872
2873 startup_mes:  DB          "Screen Editor for S-OS SWORD",__CR
2874               DB          "FE version 1.00",__CR,__CR,0
2875
2876 mes_1:        DB          "New file.",__CR,0
2877 mes_2:        DB          "Now loading ",0
2878 mes_3:        DB          "Now saving ",0
2879 mes_4:        DB          "(S)ave / (N)ot save / (C)ancel ? ",0
2880 mes_5:        DB          "Hit any key. ",0
2881 mes_6:        DB          "Memory is full.",__CR,0
2882 mes_7:        DB          "(E)dit / (M)ore find / (C)ommand",0
2883 mes_8:        DB          "(R)eplace / (S)kip / (A)ll / (C)ommand",0
2884
2885 free_colum:   DB          "[FREE:     ]",0
2886 pointer_colum:            DB     "[PNTR:   :     ]",0
2887 mark_colum:   DB          "[MARK:  :     ]",0
2888 colum_clr:    DB          "                ",0
2889
2890 s_overtype:   DB          "OVERTYPE  ",0
2891 s_insert:     DB          "INSERT    ",0
2892 s_escape:     DB          "ESC       ",0
2893 s_line_over:  DB          "LINE OVER!",0
2894 s_mem_over:   DB          "MEM OVER!!",0
2895 s_command:    DB          "COMMAND >>",0
2896
2897 tab_width:    DB          8
2898
2899 chr_TAB_swap: DB          " "
2900 chr_CR_swap:  DB          " "
2901
2902
2903 skip_character_number:
2904          DB    25
2905 skip_character_table:
2906          DB    009H,020H,022H,027H,028H,029H,02AH,02BH
2907          DB    02CH,02DH,02FH,03AH,03BH,03CH,03DH,03EH
2908          DB    05BH,05DH,07BH,07EH,0A1H,0A2H,0A3H,0A4H
2909          DB    0A5H
2910
2911
```

```
41A8 CD BF 41     2808             CALL    print_number
41AB 53           2809             LD      D,E
41AC 1E 0A        2810             LD      E,10
41AE CD 4A 41     2811             CALL    division8
41B1 5F           2812             LD      E,A
41B2 7A           2813             LD      A,D
41B3 CD BF 41     2814             CALL    print_number
41B6 7B           2815             LD      A,E
41B7 C6 30        2816             ADD     A,30H
41B9 CD F4 1F     2817             CALL    _PRINT
41BC F1           2818             POP     AF
41BD D1           2819             POP     DE
41BE C9           2820             RET
41BF              2821
41BF              2822
41BF              2823    print_number:
41BF C5           2824             PUSH    BC
41C0 47           2825             LD      B,A
41C1 FE 00        2826             CP      0
41C3 20 0E        2827             JR      NZ,prt_nl
41C5 DD 7E 00     2828             LD      A,(IX)
41C8 FE 00        2829             CP      0
41CA 20 07        2830             JR      NZ,prt_nl
41CC 3E 20        2831             LD      A," "
41CE CD F4 1F     2832             CALL    _PRINT
41D1 C1           2833             POP     BC
41D2 C9           2834             RET
41D3 78           2835    prt_nl:  LD      A,B
41D4 C6 30        2836             ADD     A,30H
41D6 CD F4 1F     2837             CALL    _PRINT
41D9 3E 01        2838             LD      A,1
41DB DD 77 00     2839             LD      (IX),A
41DE C1           2840             POP     BC
41DF C9           2841             RET
41E0              2842
41E0              2843    ; HEXCUL
41E0              2844    ;
41E0              2845    get_number:
41E0 21 00 00     2846             LD      HL,0
41E3 1A           2847    g_nl:    LD      A,(DE)
41E4 D6 30        2848             SUB     30H
41E6 FE 0A        2849             CP      0AH
41E8 D0           2850             RET     NC
41E9 13           2851             INC     DE
41EA 29           2852             ADD     HL,HL
41EB D8           2853             RET     C
41EC 44           2854             LD      B,H
41ED 4D           2855             LD      C,L
41EE 29           2856             ADD     HL,HL
41EF D8           2857             RET     C
41F0 29           2858             ADD     HL,HL
41F1 D8           2859             RET     C
41F2 09           2860             ADD     HL,BC
41F3 D8           2861             RET     C
41F4 06 00        2862             LD      B,0
41F6 4F           2863             LD      C,A
41F7 09           2864             ADD     HL,BC
41F8 30 E9        2865             JR      NC,g_nl
41FA C9           2866             RET
41FB              2867
41FB              2868
41FB              2869    ;-------------------------------------
41FB              2870    ; Data area
41FB              2871
4376              2912    control_code_table:
4376 CE 32        2913             DW      c_null                  ; 0
4378 31 33        2914             DW      c_right                 ; 1
437A 52 33        2915             DW      c_left                  ; 2
437C E1 32        2916             DW      c_up                    ; 3
437E F0 32        2917             DW      c_down                  ; 4
4380 0B 33        2918             DW      c_up_page               ; 5
4382 1E 33        2919             DW      c_down_page             ; 6
4384 8B 33        2920             DW      c_escape                ; 7
4386 D1 32        2921             DW      c_quit                  ; 8
4388 C0 33        2922             DW      c_change_typemode       ; 9
438A D1 33        2923             DW      c_tab_hid               ; 10
438C 79 33        2924             DW      c_skip_foward           ; 11
438E 82 33        2925             DW      c_skip_back             ; 12
4390 07 34        2926             DW      c_tab                   ; 13
4392 0F 34        2927             DW      c_back_space            ; 14
4394 B1 34        2928             DW      c_cr                    ; 15
4396 BD 34        2929             DW      c_block_delete          ; 16
4398 E3 34        2930             DW      c_block_cut             ; 17
439A 12 35        2931             DW      c_block_copy            ; 18
```

```
439C 37 35        2932             DW      c_block_paste           ; 19
439E EC 33        2933             DW      c_cr_hid                ; 20
43A0 BD 35        2934             DW      c_command_mode          ; 21
43A2 92 35        2935             DW      c_tab_width_change      ; 22
43A4 FF 32        2936             DW      c_cursor_top            ; 23
43A6 05 33        2937             DW      c_cursor_botom          ; 24
43A8 CE 32        2938             DW      c_reserved              ; 25
43AA CE 32        2939             DW      c_reserved              ; 26
43AC CE 32        2940             DW      c_reserved              ; 27
43AE CE 32        2941             DW      c_reserved              ; 28
43B0 CE 32        2942             DW      c_reserved              ; 29
43B2 CE 32        2943             DW      c_reserved              ; 30
43B4 CE 32        2944             DW      c_reserved              ; 31
43B6 CE 32        2945             DW      c_reserved              ; 32
43B8              2946
43B8              2947
43B8              2948    control_code_table_cmd:
43B8 CE 32        2949             DW      c_null                  ; 0
43BA 4C 33        2950             DW      c_right_cmd             ; 1
43BC 73 33        2951             DW      c_left_cmd              ; 2
43BE CE 32        2952             DW      c_null                  ; 3
43C0 CE 32        2953             DW      c_null                  ; 4
43C2 CE 32        2954             DW      c_null                  ; 5
43C4 CE 32        2955             DW      c_null                  ; 6
43C6 AA 33        2956             DW      c_escape_cmd            ; 7
43C8 CE 32        2957             DW      c_null                  ; 8
43CA C0 33        2958             DW      c_change_typemode       ; 9
43CC CE 32        2959             DW      c_null                  ; 10
43CE CE 32        2960             DW      c_null                  ; 11
43D0 CE 32        2961             DW      c_null                  ; 12
43D2 07 34        2962             DW      c_tab                   ; 13
43D4 37 34        2963             DW      c_back_space_cmd        ; 14
43D6 FF 35        2964             DW      c_cr_cmd                ; 15
43D8 CE 32        2965             DW      c_null                  ; 16
43DA CE 32        2966             DW      c_null                  ; 17
43DC CE 32        2967             DW      c_null                  ; 18
43DE CE 32        2968             DW      c_null                  ; 19
43E0 CE 32        2969             DW      c_null                  ; 20
43E2 CE 32        2970             DW      c_reserved              ; 21
43E4 CE 32        2971             DW      c_reserved              ; 22
43E6 CE 32        2972             DW      c_reserved              ; 23
43E8 CE 32        2973             DW      c_reserved              ; 24
43EA CE 32        2974             DW      c_reserved              ; 25
43EC CE 32        2975             DW      c_reserved              ; 26
43EE CE 32        2976             DW      c_reserved              ; 27
43F0 CE 32        2977             DW      c_reserved              ; 28
43F2 CE 32        2978             DW      c_reserved              ; 29
43F4 CE 32        2979             DW      c_reserved              ; 30
43F6 CE 32        2980             DW      c_reserved              ; 31
43F8 CE 32        2981             DW      c_reserved              ; 32
43FA              2982
43FA              2983
43FA              2984    END_OF_PROGRAM:
43FA              2985
43FA              2986
43FA              2987             DS      WORK_AREA - END_OF_PROGRAM
4580              2988
4580              2989
4580              2990    ;-------------------------------------
4580              2991    ; Work area
4580              2992
4580              2993
4580              2994             ORG     4580H
4580              2995
4580              2996
4580              2997    WORK_AREA:
4580              2998
4580              2999
4580              3000    mem_min:        DS      2
4582              3001    mem_max:        DS      2
4584              3002    text_start:     DS      2
4586              3003    text_end:       DS      2
4588              3004    text_max:       DS      2
458A              3005    text_size:      DS      2
458C              3006    text_free:      DS      2
458E              3007    text_area:      DS      2
4590              3008    ;
4590              3009    edited_file:    DS      1
4591              3010    ;
4591              3011    console_x:      DS      1
4592              3012    console_y:      DS      1
4593              3013    console_width:  DS      1
4594              3014    console_length: DS      1
4595              3015    ;
4595              3016    v_free_xy:      DS      2
4597              3017    v_pointer_xy:   DS      2
4599              3018    v_mark_xy:      DS      2
459B              3019    v_fnam_xy:      DS      2
459D              3020    v_mode_xy:      DS      2
459F              3021    ;
459F              3022    free_xy:        DS      2
45A1              3023    pointer_x_xy:   DS      2
45A3              3024    pointer_line_xy:DS      2
45A5              3025    mark_x_xy:      DS      2
45A7              3026    mark_line_xy:   DS      2
45A9              3027    ;
45A9              3028    view_offset:    DS      1
45AA              3029    view_line:      DS      2
45AC              3030    view_adr:       DS      2
45AE              3031    pointer_x:      DS      1
45AF              3032    pointer_line:   DS      2
45B1              3033    pointer_adr:    DS      2
45B3              3034    cursor_x:       DS      1
45B4              3035    cursor_x_keep:  DS      1
45B5              3036    locate_xy:      DS      2
45B7              3037    ;
45B7              3038    u_pointer_x:    DS      1
45B8              3039    u_pointer_line: DS      2
45BA              3040    u_pointer_adr:  DS      2
45BC              3041    u_cursor_x:     DS      1
45BD              3042    ;
45BD              3043    flag_mark:      DS      1
45BE              3044    mark_1_x:       DS      1
45BF              3045    mark_1_cr_x:    DS      1
45C0              3046    mark_1_line:    DS      2
45C2              3047    mark_1_adr:     DS      2
45C4              3048    mark_2_x:       DS      1
45C5              3049    mark_2_cr_x:    DS      1
45C6              3050    mark_2_line:    DS      2
45C8              3051    mark_2_adr:     DS      2
45CA              3052    ;
45CA              3053    flag_cmd_mode:  DS      1
45CB              3054    ;
45CB              3055    copy_length:    DS      2
45CD              3056    flag_cant_past: DS      1
45CE              3057    ;
```

▶Xellent30s, HARPやクロックアップよりスマートでまずまずの性能でいいと思います。古いマシンを愛着をもってまだまだ使ってあげられそうです。

栗林　幹治(22)千葉県

```
45CE                 3058  line_length:     DS       1
45CF                 3059  flag_edited:     DS       1
45D0                 3060  edited_length:   DS       1
45D1                 3061  ;
45D1                 3062  line_print_y:    DS       1
45D2                 3063  line_data_adr:   DS       2
45D4                 3064  chr_atri:        DS       1
45D5                 3065  ;
45D5                 3066  type_mode:       DS       1
45D6                 3067  escape_mode:     DS       1
45D7                 3068  ;
45D7                 3069  undo_l_b_adr:    DS       2
45D9                 3070  find_b_adr:      DS       2
45DB                 3071  replace_b_adr:   DS       2
45DD                 3072  copy_b_adr:      DS       2
45DF                 3073  buffer_size:     DS       2
45E1                 3074  ;
45E1                 3075  flag_replace_all:
45E1                 3076                   DS       1
45E2                 3077  find_str_len:    DS       1
45E3                 3078  replace_str_len:DS       1
45E4                 3079  ;
45E4                 3080  local_work:      DS       4
45E8                 3081  filename:        DS       20
45FC                 3082  tab_info_table:  DS       256
46FC                 3083  line_buffer:     DS       256
47FC                 3084                   DS       4
4800                 3085
4800                 3086
4800                 3087
4800                 3088  ; FE version 1.00
4800                 3089  ; MAY 21,1995. By H.Matsufuji
```

リスト5 FELP.ASM

```
0000                 1  ;---------------------------------
0000                 2  ;
0000                 3  ;  line_print routine for FE v1.04
0000                 4  ;  for general
0000                 5  ;
0000                 6  ;---------------------------------
0000                 7
0000                 8
1FF4 P               9  _PRINT  EQU     1FF4H
201E P              10  _LOC    EQU     201EH
0000                11
0000                12
3069 P              13  next_chr           EQU     3069H
0000                14
4591 P              15  console_x          EQU     4591H
4592 P              16  console_y          EQU     4592H
4593 P              17  console_width      EQU     4593H
4594 P              18  console_length     EQU     4594H
0000                19
45D2 P              20  line_data_adr      EQU     45D2H
45D1 P              21  line_print_y       EQU     45D1H
45A9 P              22  view_offset        EQU     45A9H
45FC P              23  tab_info_table     EQU     45FCH
0000                24
3016 P              25  nm_chr_atri        EQU     3016H
45D4 P              26  chr_atri           EQU     45D4H
0000                27
0000                28
0000                29          OFFSET  0C000H-4400H
4400                30          ORG     4400H
4400                31
4400                32
4400                33  line_print:
4400 C5             34          PUSH    BC
4401 D5             35          PUSH    DE
4402 E5             36          PUSH    HL
4403 FD E5          37          PUSH    IY
4405 DD 5E 00       38          LD      E,(IX)
4408 CD 3C 44       39          CALL    locate_init
440B 3A 16 30       40          LD      A,(nm_chr_atri)
440E 32 D4 45       41          LD      (chr_atri),A
4411 2A D2 45       42          LD      HL,(line_data_adr)
4414 FD 21 FC 45    43          LD      IY,tab_info_table
4418                44  ;
4418 3A A9 45       45          LD      A,(view_offset)
441B 57             46          LD      D,A
441C 14             47          INC     D
441D 15             48  lp1:    DEC     D
441E CA 27 44       49          JP      Z,lp2
4421 CD 69 30       50          CALL    next_chr
4424 C3 1D 44       51          JP      lp1
4427                52  ;
4427 3A 93 45       53  lp2:    LD      A,(console_width)
442A 57             54          LD      D,A
442B 15             55          DEC     D
442C CD 69 30       56  lp3:    CALL    next_chr
442F                57  ;
442F CD F4 1F       58          CALL    _PRINT
4432                59  ;
4432 15             60          DEC     D
4433 C2 2C 44       61          JP      NZ,lp3
4436 FD E1          62          POP     IY
4438 E1             63          POP     HL
4439 D1             64          POP     DE
443A C1             65          POP     BC
443B C9             66          RET
443C                67
443C                68
443C                69  locate_init:
443C 3A D1 45       70          LD      A,(line_print_y)
443F 67             71          LD      H,A
4440 3A 92 45       72          LD      A,(console_y)
4443 84             73          ADD     A,H
4444 3A 91 45       74          LD      A,(console_x)
4447 6F             75          LD      L,A
4448                76  ;
4448 CD 1E 20       77          CALL    _LOC
444B C9             78          RET
444C                79
444C                80
```

リスト6 FELP_X1.ASM

```
0000                 1  ;---------------------------------
0000                 2  ;
0000                 3  ;  line_print routine for FE v1.05
0000                 4  ;  for X1/X1turbo non Kanji
0000                 5  ;
0000                 6  ;---------------------------------
0000                 7
0000                 8
1F5C P               9  _WIDTH  EQU     1F5CH
0000                10
0000                11
3069 P              12  next_chr           EQU     3069H
0000                13
4591 P              14  console_x          EQU     4591H
4592 P              15  console_y          EQU     4592H
4593 P              16  console_width      EQU     4593H
4594 P              17  console_length     EQU     4594H
0000                18
45D2 P              19  line_data_adr      EQU     45D2H
45D1 P              20  line_print_y       EQU     45D1H
45A9 P              21  view_offset        EQU     45A9H
45FC P              22  tab_info_table     EQU     45FCH
0000                23
3016 P              24  nm_chr_atri        EQU     3016H
45D4 P              25  chr_atri           EQU     45D4H
0000                26
0000                27
0000                28          OFFSET  0C000H-4400H
4400                29          ORG     4400H
4400                30
4400                31
4400                32  line_print:
4400 C5             33          PUSH    BC
4401 D5             34          PUSH    DE
4402 E5             35          PUSH    HL
4403 FD E5          36          PUSH    IY
4405 DD 5E 00       37          LD      E,(IX)
4408 CD 4C 44       38          CALL    locate_init
440B 3A 16 30       39          LD      A,(nm_chr_atri)
440E 32 D4 45       40          LD      (chr_atri),A
4411 2A D2 45       41          LD      HL,(line_data_adr)
4414 FD 21 FC 45    42          LD      IY,tab_info_table
4418                43  ;
4418 3A A9 45       44          LD      A,(view_offset)
441B 57             45          LD      D,A
441C 14             46          INC     D
441D 15             47  lp1:    DEC     D
441E CA 27 44       48          JP      Z,lp2
4421 CD 69 30       49          CALL    next_chr
4424 C3 1D 44       50          JP      lp1
4427                51  ;
4427 3A 93 45       52  lp2:    LD      A,(console_width)
442A 57             53          LD      D,A
442B 15             54          DEC     D
442C CD 69 30       55  lp3:    CALL    next_chr
442F                56  ;
442F ED 79          57          OUT     (C),A
4431 CB A0          58          RES     4,B
4433 3A D4 45       59          LD      A,(chr_atri)
4436 ED 79          60          OUT     (C),A
4438 CB E0          61          SET     4,B
443A CB D8          62          SET     3,B
443C AF             63          XOR     A
443D ED 79          64          OUT     (C),A
443F CB 98          65          RES     3,B
4441 03             66          INC     BC
4442                67  ;
4442 15             68          DEC     D
4443 C2 2C 44       69          JP      NZ,lp3
4446 FD E1          70          POP     IY
4448 E1             71          POP     HL
4449 D1             72          POP     DE
444A C1             73          POP     BC
444B C9             74          RET
444C                75
444C                76
444C                77  locate_init:
444C 3A D1 45       78          LD      A,(line_print_y)
444F 67             79          LD      H,A
4450 3A 92 45       80          LD      A,(console_y)
4453 84             81          ADD     A,H
4454 3A 91 45       82          LD      A,(console_x)
4457 6F             83          LD      L,A
4458                84  ;
4458 4D             85          LD      C,L
4459 7C             86          LD      A,H
445A 87             87          ADD     A,A
445B 87             88          ADD     A,A
445C 84             89          ADD     A,H
445D 26 00          90          LD      H,0
445F 6F             91          LD      L,A
4460 29             92          ADD     HL,HL
4461 29             93          ADD     HL,HL
4462 29             94          ADD     HL,HL
4463 3A 5C 1F       95          LD      A,(_WIDTH)
4466 FE 50          96          CP      80
4468 20 01          97          JR      NZ,loc_i1
446A 29             98          ADD     HL,HL
446B 06 30          99  loc_i1: LD      B,30H
446D 09            100          ADD     HL,BC
446E 44            101          LD      B,H
446F 4D            102          LD      C,L
4470 C9            103          RET
4471               104
4471               105
```

▶おかげさまで，もう一息で福岡は夜間断水が解除になりそうです。でも……今年は冷夏の予感……。
松尾　繁(21)福岡県

軽やかで重い電子郵便の世界

電子メールのきっかけはwww

イタリアの学生から電子メール(以下メール)が届きました。先月号で紹介したように，インターネット上にwwwで自分のホームページを即席で作ったところ，それを見て問い合わせてきたのです。いくつかの自分の論文についてその概略を英文で載せておいたので，そのうちのひとつに興味をもった学生がその論文の入手法を聞いてきたのでした。

自分としては，サービス精神を発揮して作ったところがちょっぴりあるものの，まだまだ非公式なものであり，wwwとはいったいどんなものかという興味だけで作ってみたという段階でした。もちろん，「世界に情報を発信してやるぞ」などという意気込みは皆無です。

それにしても，なにかいいネタはないかと探しまくっている人は世界中にワンサカ(死語？)いるみたいです。作ったばかりのホームページなのに，あっという間に探し出してしまうのですから。

大学のデータから順にたどってきたのではなく，論文のデータが入っているページにいきなりたどり着く人はこのイタリア人以外に毎日何人もいます。どうやって，このMacintoshの中のデータまで達するのかは，だいたいわかっています。ワシントン大学にものすごいデータベースがあり，このページの中にキーワードが登録されているので，そこで検索すると僕のページもリストアップされるのです。

でも，どうやってそのデータベースに僕のページが登録されたのか？　という謎があります。自分がやったことといえば，部屋のMacintoshでMacHTTPというwwwサーバプログラムを起動して，ほぼ24時間稼動させていること，Httlという言語で講座と自分のデータを記述したこと，それからうちの学部のページの中からリンクをはってもらった(自分の講座名をクリックすると自分のMacintoshに飛ぶような記述を加えてもらった)ことだけなのです。

むろん，各大学の各学部の各講座の各人のホームページをしらみつぶしに調べている人がいれば可能ですが，ちゃんとしたデータベースを作っているのならば，ひとりの手では無理ですから，組織的にやっているか，あるいはソフトウェアで自動的に探索しているかのどちらかでしょう。単にキーワードを含むページのアドレスだけをため込んでいるのでしょうが，世界中のデータですからやはり膨大であろうと予想されます。

実際はどうなっているのか？　説明がそのデータベースのところに書かれていました。やはり，プログラム(複数のエージェント)でどんどんリンクをたどってはキーワードを登録しているとのことです。リンクをどこから，たどりはじめるのかということについては，特に示されていませんが，そういうリンクを集めた有名なページなどからのようです。

電子メール普及運動

この連載の最後にときどき僕のメールアドレスを載せてもらうようにしていますが，少なからず反応があります。特に，5月号の「計算機の中の『やらせ』問題」のときは，興味をもってくださった方からのメールが自分が本誌を手にする前から何通もきて，それなりの手ごたえを感じました。

以前にも載せてもらったことが数回ありますが，そのときよりメールが増えているので，メールも普及してきたのでしょう。そして今後もいっそう普及していくことでしょう。なんといっても便利ですから。

メールの普及には微力ながら役立ちたいと思っています。文系の学生向けの授業の中でも何度か計算機を触ってもらっていますが，メールやニュースで遊ぶこともその重要なテーマにしています。

「計算する機械」としての計算機よりは，「接続する機械」としての計算機のほうが，今後は我々の生活や内面により直接的に影響していくのではないかと思っています。そして，その第一歩としてメールを体験してもらおうとしているわけです。

今年度も，まずUNIXのワークステーションの実習がありまして，ログインの仕方，ウィンドウの操作方法，マウスやキーボードの使い方，そしてメールの使い方を学んでもらいました。

実習の最後に僕宛に感想などを送ってもらったりするのですが，「メールを出すことがこんなに面白いことだとは思わなかった」などという意見が多いのです。ログイン，漢字変換，メールの送信までを1コマ(1時間半)でこなしてしまうのですから，いやはや電脳世代ですね(ちょっとハードすぎると反省してますが)。

メールの基本を押さえる

メールのメディアとしての性格について基本だけ押さえることにしましょう。一般に，人と人が1対1でコミュニケーションをとる方法として通常行われるのは，電話，手紙ですね。それと，メディアを通さないで直接面と向かってというのも比較の対象として入れることにしましょう。これらを4つの観点からメールと比較して表にしました。

	即時性	記録性	情報量	開放性
対面	○	×	○	×
電話	○	×	△	△
手紙	×	○	△	△
メール	□	○	×	○

●即時性

即時性とは，同時双方向的にやりとりができるかどうかということです。面と向かっての会話とか電話はむろんリアルタイムそのものです。手紙は×です。メールは状況により，一概にいえません。

ネットワーク的に近い距離にある2者間では瞬時に相手とのやりとりができますので，それこそ話し言葉そのもののようなやりとりが可能です。たとえば，次のようにメールが素早くやりとりされます。

［メール1］
ところでこのあいだの一件どうなった
？
［メール2］
>ところでこのあいだの一件どうなっ
た？
「このあいだの一件」って，例のAさん
のこと？
［メール3］
そうです。
［メール4］
本人が恥ずかしがってなんともいわな
いんだよ。
めでたい話なのにね :-)

illustration：Haruhisa Yamada

まるで口語です。ただし，メールならではという文章スタイルや表記法なども生まれてきています。たとえば，相手のメール中の文章を引用することはごく一般的です。例では，メール2で引用しています。また，メール4の最後にあるような文字を使った絵もいろいろ考えられています(これは顔を左に90度傾けるという情報です)。

先の表で，メールのリアルタイム性について□マークをつけたのは，このようなリアルタイムの会話が(物理的にも許されればですが)可能であると同時に，意識的にリアルタイム性を拒絶して手紙のような遅い応答を選択することもできるということによります。

つまり，相手からのメールに対して，即座に反応しなくても，特に問題がない場合が多いというわけです。ですから，リアルタイム性に関しては，どちらでも自由に選択できるという意味で○とは違う□マークをつけました。

電話は相手がいないと会話が成り立ちませんし，いるならば相手の時間を電話の会話に即座に費やしてもらう必要が生じます。メールならば，時間が空いたときに応答すればよく，それが誰にでも気軽に出しやすいということにつながります。

日常の話し言葉を文体としてはとりながらも，実はその言葉を発するまでに自由に時間をかけることが可能であるということ，この点からいってもきわめてユニークな文体が生まれつつあると思います。気軽な言葉であっても実は数時間にも及ぶ熟慮の結果かもしれないからです。

●情報量，記録性，開放性

情報量の観点から見た分類は容易です。面と向かえば膨大な情報(主に聴覚と視覚からですが)を得ることができます。手紙になると文字情報あるいは図情報を送ることができます(ビデオテープを同封したらなどというのはありますが)。さらに電話になると音声情報のみに制限されます(これも最近はテレビ電話が登場していますが)。

基本的なメールでは文字列のみということになります。つまり，情報量の制限が大きくなるわけです。だから，先ほどのスマイルマークなどが発展するともいえます。

ただし，文字列のみといっても，紙に書かれた文字とは根本的に違うところがあります。それは，メールの場合，その文字たちは画面上を流れていく(スクロールする)という点です。流れていく文字ということが，先に指摘したような軽やかな口語調の文体を実現してきたひとつの土台となっているとみなすのは考えすぎでしょうか？

手紙のように紙という形式ではありませんが，ファイルという形でがっちりと記録されるわけです。また，好きなときにプリントアウトすることもできますが，画面上を軽やかに流れるということが実はがちがちの文章を排除する方向へと進めている力になっていると考えるのです。

実際，メールのこの軽やかな会話体に慣れていくと，しだいに細かなミスタイプだとか，ひとつの文書の中の細かい矛盾などは無視するようなおおらかさが身についていくのではないでしょうか。

さて，情報量が制限されるということはメディアとして，直接的にはマイナス要因なわけですが，メールを使っていると，情報量の制限ということが実は予想もしないような性質につながっていることに気がつくようになります。

まず，相手との関係が率直で対等なものになりやすいということが挙げられます。相手がどんな格好をしたどんな身分のどんな怖そうなおっさんかは知らずに，それなりにコミュニケーションすることができるのですから。会った こともない人と堅苦

軽やかで重い電子郵便の世界

しくなく気楽にやりとりができるという開放性に関してはメールは抜きん出ていると思います。

さらにいうならば，メールを重要なメディアとする計算機ネットワーク上の社会には，実際の所属とか身分などにとらわれない自由なコミュニケーションを実現したいという参加者の願望に関するコンセンサス(合意)はかなりできあがっているのではないかと思います。そもそも，計算機の中にできあがっているせっかくの別の世界ですから，実世界のしがらみなどを持ち込むのはヤボということにしましょう。

また，一方で，情報量が少ないことは，読み手に無意識にほかの情報を行間に読むという習性をつけさせることにもなります。よく，テレビに比べて本は想像力をかきたてるといいますが，それと同じことです。顔色を見れば，あるいは声の調子を聞けばすぐわかるようなことを，文章ににじみ出る雰囲気から読み取ることになります。

万々歳でもない？

メールの特性を整理してその影響やさまざまな相互関係をきっちりまとめるのはかなり難しいことのようです。ほかのメディアとの簡単な比較をもとに多少解析したわけですが，まだまだもの足りません。

そして，メールの特性が生む矛盾についても考えさせられます。2つほど書いておきます。

●平等性と特権性

情報量の制限，開放性などから，平等な関係が作り出されるということを書いたのですが，それ以前の問題があります。それは，メールが便利であればあるほど，それを使える人と使えない人との格差が生じるという事実です。環境的に使えない人はどうしても使えないわけです。

この点は教育とか政治とかに期待せざるを得ないのでしょう。この文章が「絶望」ということを意味していないことを期待したいものです。

●軽い文体と記録性

気軽に書くことができるということ，誰もが平等であるということが一方にありますが，他方では情報量が少なく，しかも記録に残るということがあります。この2つの性質の対立が問題です。

ついうっかりとまずいことを書くと，おしゃべりとは違って，のちのちにも残ります。気が変わっても記録が残っているとそれは矛盾になります。もともと，行間からいろいろ読もうとする習性がついているので，いったんもつれ始めると解釈が悪いほう悪いほうにと進み，泥沼状態になりやすいわけです。気をつけましょう。

データは物語る

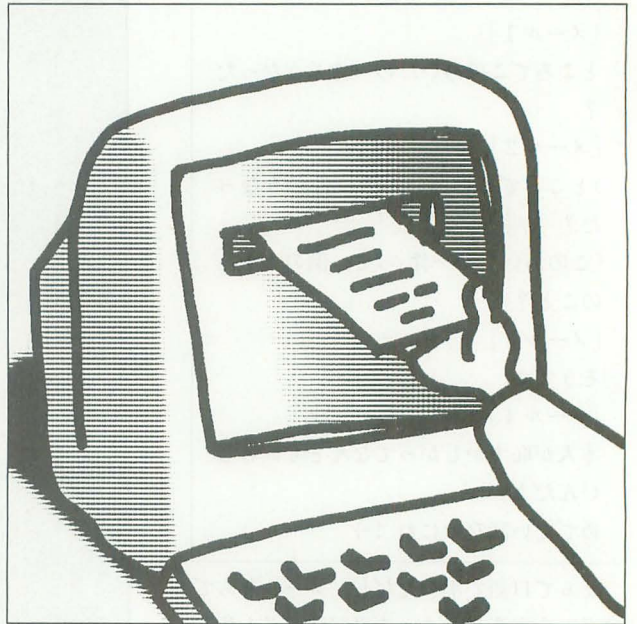
実際のところ，メールを使うとどのようなご利益(りやく)が得られるのか？　あるいは損をするのか？，研究もさまざまになされているようです。雑誌『ワイアード』に興味深いデータが載っていたのでいくつか紹介しましょう。

●企業内で男女が対面した会議の場合，意思決定に関わる発言を最初にする回数は男性が女性の5倍であるが，メールによる会議では男女の比率が同数になった。

「平等がもたらされたという理想的なデータですね」

●メールを導入した大手保険会社では42%の役員がメールによる意思伝達がベストであると答えた。対面がよいと答えた人々は32%に留まった。

「若い役員の多い会社なのでしょうね。日本の大手企業じゃあ，まずありえない数字です。居酒屋でコミュニケーションですからね，なんたって(自分も好きですが)」

●メールを頻繁に利用している科学者ほど活動的かつ生産的である。

「まあ，これは事実でしょう。活動的とか，生産的というのは，必ずしも，その研究の質に直接影響しませんしね。計算機を活用できているということだけでも，まあ活動的だといえますしね」

●お互いの反応を見ることも聞くこともできないので偏見のない会話が成り立つ。上役に対しても遠慮なく下品なことばを使い，さらには悪口の応酬(フレーミング)さえ生じやすくなる。

「これは，先に述べてきたとおりですね」

そろそろ，終わりが近づいてきたので，珍しくもまとめをば。

・画面上を流れる軽やかさを楽しもう。でも，じっくり書いて，じっくり読もう！

・いろいろなメディアをうまく使い分けよう！

・電子郵便局を国が作り，小学校で使い方を教えて，すべての人が自由にメールを使えるようにしよう！

参考文献

ジャック・レスリー，「電子メイルは魔法？」，ワイアード，1995.Vol.1 No.02，98-101pp.

e-mailアドレス

ari@info.human.nagoya-u.ac.jp

NIFTY-ServeやPC VANから送信するときは，前者がINET:後者がINET#を上記のアドレスの前につける。

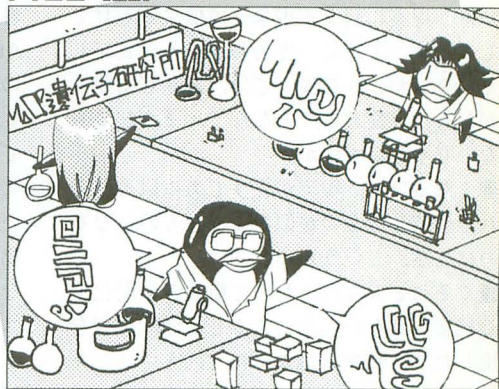
こちらシステムX探偵事務所

計算モデルの動的割り当て

Shibata Atsushi 柴田 淳

FILE-XXIV

illustration：T. Takahashi

第3世代のシミュレーションを模索していた柴田氏ですが，なんとなく解決の手法が見えてきたようです。今回はその手法となる「計算モデルの動的割り当て」について説明していきます。サンプルプログラムにもこの手法が用いられています。

シムシティー型の問題点

マスター（以下M）：前回，第3世代の社会科学系シミュレーションのことが出てきましたよね。そのあたりのアイデアを，もうちょっと詳しく聞かせてくださいよ。

琴張護（以下護）：確か，シミュレーションの際に使われる計算モデルを「動的」に割り当てるという話ではなかったでしょうか？

柴田淳（以下Ats）：ええと，そもそもこのアイデアを思いついたのは，シムシティーのようなゲームの欠点を，なんとか取り除くことができないか，と考えていたときのことなんです。

琴張春香（以下春）：欠点というと？

Ats：シムシティーって，ある程度街を育てて，マップが建物で埋まってしまうと，とたんに面倒臭くなるでしょう。

M：確かに，面倒臭さを感じてしまったら最後，もうそれ以上ゲームを続ける気がしなくなりますよね。

Ats：仮に，プレイヤーの操作を入力，操作の結果として画面に現れる変化を出力とすると，入力に対してどれだけ大きな出力を得られるかが，ゲームの面白さを決める重要な要素である，といえるんじゃないでしょうか。で，シムシティー場合，適当に道路や発電所などの公共施設を建設すれば，どんどん街が育っていくわけです。

M：つまり，少ない入力で大きな出力を得ることができるわけですね。

Ats：ところが，マップ中が建物で埋まってしまうと，プレイヤーの入力が画面の変化に対して与える影響が，極端に小さくなってしまうんです。

春：なるほどね。街が育ってしまうと，少しくらい道路を敷き直しても街並みはそう変わらなくなるから，プレイヤーはたくさんの入力を強いられる，というわけね。

護：しかし，そのような現象が起こる原因はどこにあるのでしょう？

Ats：裏マップの動きを考えてみればわかると思います。

M：裏マップって，マップの格子ごとに，地価や公害汚染度などを記憶しておくためのものでしたよね。

Ats：この裏マップは，ゲームを始めた時点ではまっさらですから，ちょっと道路を敷いただけでも，値は大きく変わります。しかし，ゲームが進んで裏マップの数値が大きくなったあとでは，少しくらい道路を敷き直しても，数値は大きくは変わらなくなってしまうんです。

春：1カ月の小遣いが1,000円から1,500円に上がったらうれしいけど，10,000円から10,500円に上がってもそれほどうれしくないのと似たようなものね。

M：……なんとも即物的な比喩ですね。

Ats：でも，ちゃんと的を射ていますよ。要するに，数値の増加が同じでも，元の数値が大きければ画面に現れる変化は小さい，ということが問題なんです。

護：そういえば，これまでのサンプルプログラムでも，実行してからある程度時間が経つと，急に画面に変化が現れなくなるものがほとんどでしたが，これも同じことが原因になっているのでしょうか。

Ats：そうだと思います。シミュレーションの計算モデルには，ほとんどの場合重加算モデルというものが使われていると，以前に話したと思います。重加算モデルというのは，単純に1次式を組み合わせたものですから，ゲームが進んで入力される値が大きくなれば，出力される値は，比率で見る限り大きく変化しなくなるのは当然ですよね。

護：では，計算モデルに使う数式として，重加算モデル以外の数式を採用すればいいのではないでしょうか。たとえば，指数関数のように入力される変数が大きくなればなるほど，出力される値が急激に大きくなっていくような数式はどうでしょう？

Ats：いや，それは本末転倒ですよ。だって，社会現象などを数式で表すとほとんどの場合重加算モデルになる，という前提があるからこそ，1次式の組み合わせが使われるんですから。「SIMねずみ講」とかいうゲームを作るとしたら，ひとつくらい指数関数を使うかもしれないけど。

春：それなら，シムシティー型のゲームでは，時間が経っても画面の変化が途絶えないようなゲームを作るのは不可能，ということになるじゃない？

Ats：そんなことないですよ。たとえば，ある程度街の規模が大きくなったら，駅の周辺の再開発条令が公布されるようにするんです。再開発地域では，容積率が緩和されたり，補助金が出たり，つまり計算モデル自体が微妙に変化しますよね。だから，裏マップの評価のされ方も変わってくるわけです。

護：なるほど。そこで，計算モデルを動的に割り当てて，モデルの数式自体を変えていくというアイデアが生まれてくるわけですね。

Ats：ただ，計算モデルを変化させればそれでいいか，というとそうでもないんです。先ほどの例にならって，街が育って再開発条令が公布されるようにしても，公布後一定時間経つと，また以前と同じようにマップの変化が停滞してしまう可能性が高いですよね。

M：ふと思ったんですけど，シムシティーみたいに，社会現象をシミュレートするゲームでは，根本的に「状況の停滞」という問題がつきまとうんじゃないでしょうか？

春：というと？

M：だって，シミュレーションというから

には，現実に起こる現象を模倣するわけでしょう？　実際に一度街のつくりが安定してしまったような場所では，数年で街並みががらりと変わることなんて，まず起こらないじゃないですか。

春：なるほど。現実がそうである以上，それを模倣したものが似たようなものになるのは当然，というわけね。

護：それでもなんとかして街並みの変化を途絶えさせないようにすると，怪獣が街を壊したり，戦争を起こしたり，といったキワモノに走るしかないでしょう。

Ats：こうして見てみると，都市の発達シミュレーションには原理的に限界がありそうですね。

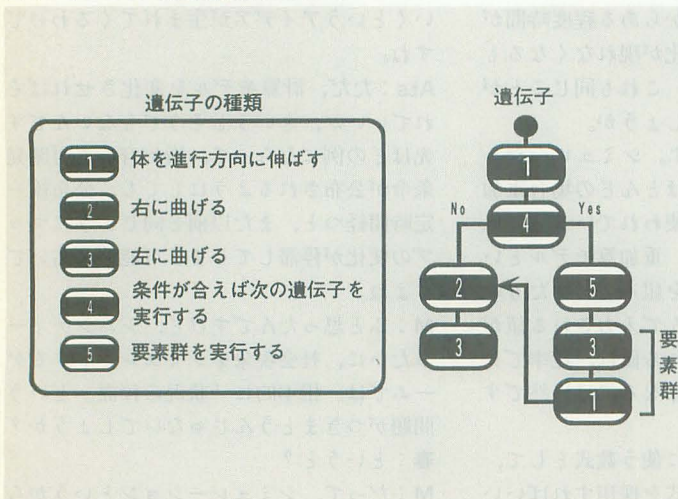
M：だとしたら，「計算モデルを動的に割り当てる」というアイデアはどうなるんですか？

Ats：都市の発達シミュレーションではなく，ほかのシミュレーションであれば，このアイデアは生かせますよ。たとえば，遺伝のシミュレーションなんてどうでしょう。

遺伝のシミュレーション

M：ところで，遺伝のシミュレーションと計算モデルを動的に割り当てるアイデアって，あまり関係がなさそうに思えるんですけど。

春：というより，そもそも計算モデルを動的に割り当てる，っていうのはどういうことなの？

Ats：ええと，普通，Cのソースを書いてコンパイルすると，関数を呼び出している部分はソースコードとしてメモリ上に埋め込まれてしまうでしょう。つまり，関数の呼び出し部分は動かないので，これを称して「静的」と呼ぶわけです。

春：じゃあ，動的の場合は？

Ats：関数の呼び出しを動的にすると，どの関数を呼び出すかを，比較的自由にすることができるんです。たとえば，とある行でAという関数を呼び出していたとすると，この呼び出し部分が静的に割り当てられていれば，そこからはAという関数しか呼び出せない。しかし，関数のアドレスを変数などに保存しておいて，その関数のアドレスを直接コールするようなソースを書くんです。

M：なるほど。そうすれば，場合によっては，BやCなどほかの関数を呼び出せる，というわけか。

護：それで，この関数の動的割り当てと遺伝のシミュレーションはどのようにつながるのですか？

Ats：そう先を急がずに。まず，自分の体をなるべく長く伸ばそうとする線虫のような生命体を想定しましょう。で，この線虫は遺伝子をもっていて，自分の遺伝子に従って体を伸ばしていくんです。

春：遺伝子に従って体を伸ばしていく？

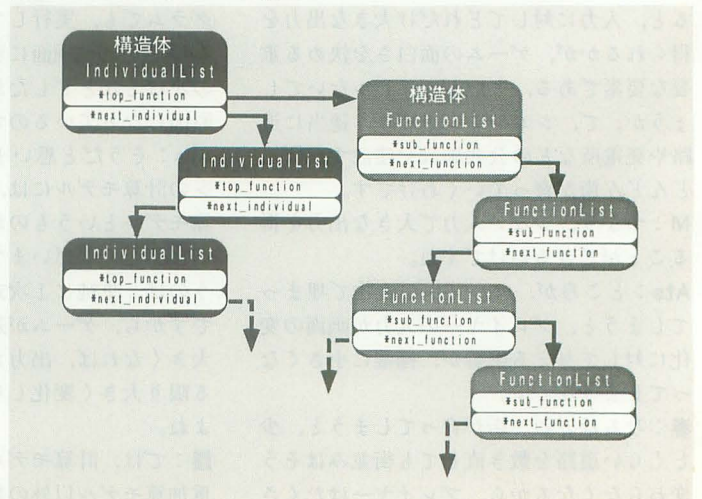
Ats：つまり，遺伝子が体を伸ばしていくための法則というか，アルゴリズムのような働きをするわけです。

M：それをプログラミングするんですか？なんだか大変そうだけど。

Ats：ただ，遺伝子といっても複雑なものではなくて，いくつかの単純な動作を実行するだけです。まずは動作の定義から。

1）　体を進行方向に伸ばす
2）　右に曲げる
3）　左に曲げる

護：これらの遺伝子の要素が順番に並んでいて，線虫はこの遺伝子に沿って体を伸ばしていくわけですね。

Ats：ただ，これだけじゃつまらないから，次のような特殊な遺伝子も考えます。

4）　条件が合えば次の遺伝子を実行する
5）　要素群を実行する

春：4番目はいいとして，5番目はどういう動作をするの？

Ats：遺伝子の中に遺伝子が入れ子になっていて，その遺伝子を実行するんです。

護：なぜそのような要素が必要なのでしょうか？

Ats：図1のように，5番目が条件分岐の直後にある場合を考えれば，その必要性がわかると思います。

M：なるほど。要素群を実行する遺伝子要素があるおかげで，条件によって複数の要素を実行したりしなかったりという場合分けが可能になるんですね。

突然変異

Ats：さて，この要素を順に4回実行して，線虫を成長させます。しかし，ただ適当に遺伝子を組み立ててもつまらないので，要素1個の単純な遺伝子から出発して，それを突然変異させて，より複雑な遺伝子を作り上げてみましょう。

春：遺伝子が突然変異するということは，遺伝子の中身が変化するのよね。

護：遺伝子の中身を変化させるには，たとえば，5つの基本動作を入れ替える方法があります。

Ats：そこで関数の動的割り当てというアイデアが生きてきます。つまり，遺伝子の動作，すなわち遺伝子の要素が呼び出す関数のアドレスを，構造体のメンバーに記録しておくんです。

M：ということは，遺伝子のとある要素の動作が突然変異によって入れ替わると，関数のアドレスを記録したメンバーが書き換

図1　サンプルの遺伝情報の基本パターン

図2　リストと呼ぶデータ構造

えられるわけですね。

護：ところで，遺伝子を突然変異させるということは，遺伝子は可変長である必要があるのではないでしょうか。

春：可変長って？

護：つまり，遺伝子の要素を保存するために配列を用意したとすると，配列を宣言した時点での添え字によって，遺伝子の長さの上限が決まってしまうのです。

M：配列の添え字を十分大きく取っておけばいいんじゃないですか？

Ats：それではさすがに無駄が多いので，遺伝子の要素の数を可変にするために，リストというデータ構造を導入します。

M：リストというと，遺伝子の要素のデータ中に，次の要素の位置を保存しておくわけですか。

Ats：そうなんです。今度は図2を見てください。まず，線虫の個体用にIndividual Listという構造体を定義してあります。この構造体の中にtop_functionというポインタがあって，これが遺伝子の最初の要素を指しているんです。

護：図によると，遺伝子の各要素の情報はFunctionListという構造体に定義されているようですね。

M：そして，構造体FunctionListのメンバーnext_functionが，次にくる遺伝子の要素を指し示している，というわけですか。

春：図2を見ると，リストって学生のころお世話になった電話の連絡網みたいね。

M：そうですね。1人ひとりの生徒は，ただ単に自分の次の生徒に電話をかければいいんですからね。リストの要素も，自分の直後の要素の位置だけを覚えておけばいいわけですし。

Ats：あと，クラスに新入生が入ってきたら，連絡網の一番最後につけ足せば，クラス全員に連絡がいきわたりますよね。それと同じ理屈で，ある遺伝子に要素をつけ足すときには，リストの一番最後に新しい要素をつけ足すだけでいいんです。

春：そうやって遺伝子の要素を増やしたり突然変異を起こしたりして，いろいろな線虫を育てていくわけね。

Ats：次に，サンプルのリストに目を移しましょう。まず，グローバル変数のIndividualsが，線虫の個体のリストの先頭を指すポインタになっています。119行目からの関数Evoluteでこの変数から各個体を抜き出して，次に遺伝子に従って，線虫の体を伸ばします。

M：線虫の体を伸ばすためには，141行目からのExecuteActionsという関数が呼ばれていますね。

Ats：遺伝子の要素もリスト構造をもっているわけですが，この要素のリストを次々に実行するのが，次にある関数Executeです。この関数は，遺伝子の要素のリストをたどっていって，リストが途絶えるまで要素の動作を実行します。これを4回繰り返して，線虫は体を伸ばすのをやめます。

春：線虫の体を伸ばしたあとはなにをするの？

Ats：関数Evoluteに戻って，次は遺伝子を突然変異させます。199行目からの関数Mutateが突然変異を受けもっています。1つの遺伝子から始めて，画面が埋まるまで個体の遺伝子をコピーして，自分自身とコピーした遺伝子に突然変異を起こします。

M：プログラムを実行してみると，最初のうちは大きな個体が育ちませんね。

護：遺伝子が小さいうちは，変化のバリエーションが少ないためループする可能性が高くなるのでしょう。遺伝子が大きくなれば，安定した個体が登場するはずです。

春：あと，突然変異の直前に個体がコピーされるわけだから，個体間には血縁関係があるわけでしょう？

M：画面を見る限り，どの線虫がどの線虫の子孫なのかがわかりづらいですね。同じ位置にある線虫は少しずつ変わっているのがわかりますけど。

Ats：そうですね。血縁関係をはっきりさせるためには，先祖と子孫が似ていなければならないわけですけど，そのためには突然変異率を抑えるとか，遺伝子の変わり方になんらかの制約を設ける必要があるわけです。

護：しかし，制約を厳しくすると，個体の変化が乏しくなります。

M：なるほど。そのあたりのバランスを取るのが難しいというわけですね。

Ats：また，このプログラムのように計算モデルを動的に割り当てる手法と，裏マップを使ったシミュレーションをどのように組み合わせるか，という問題もあります。次回はそのあたりを突き詰めてみたいと思っています。　（つづく）

リスト

```
 1: #include <stdio.h>
 2: #include <stdlib.h>
 3: #include <stddef.h>
 4: #include <string.h>
 5: #include "basic.h"
 6: #include "graph.h"
 7:
 8: #define true    1
 9: #define false   0
10:
11: typedef long        listed_func(void*,long*,long*);
12:
13: typedef struct  {
14:         long        action[50],
15:                     position,direction,length,
16:                     lengthPrev;
17: } History,*HistoryPtr,**HistoryHandle;
18:
19: typedef struct  {
20:         long        par1,par2;
21:         listed_func     *function;
22:         History *theHistory;
23:         void        *next_function,*sub_function;
24: } FunctionList,*FunctionListPtr,**FunctionListHandle;
25:
26: typedef struct  {
27:         FunctionList    *top_function;
28:         void        *next_individual;
29:         History *theHistory;
30: } IndividualList,*IndividualListPtr,
31:   **IndividualListHandle;
32:
33: enum    {
34:         up = 0,left,down,right };
35:
36: enum    {
37:         state_false = 0,state_true,state_stucked,
38:         state_no_executes,
39:         go_straight,turn_right,turn_left,
40:         free_self = 100 };
41:
42: enum    {
43:         no_err = 0,mem_err };
44:
45: enum    {
46:         Fstraight = 0,Fright,Fleft,Fcheck,Fmax };
47:
48: long    dir_x[4] = {
49:         0,-2,0,2 },
50:         dir_y[4] = {
51:         -2,0,2,0 };
52:
```

```
 53: IndividualListPtr Individuals = NULL;
 54: HistoryPtr      Histories = NULL;
 55: long       total = 1,wx,wy,ox,oy,wpos;
 56: listed_func      *Funcs[5];
 57:
 58: long     Evolute(void);
 59: void     ExecuteActions(IndividualListPtr ind);
 60: long     Execute(FunctionListPtr target,long* x,long* y);
 61: long     Mutate(IndividualListPtr source);
 62: long     MutateFunc(FunctionListPtr target);
 63: IndividualListHandle    GetLastIndividual(void);
 64: long     NewInd_his(IndividualListHandle ind,
 65:                          HistoryHandle his);
 66: long     CopyInd_his(IndividualListHandle target_ind,
 67:                   HistoryHandle target_his,
 68:                   IndividualListHandle source_ind,
 69:                   HistoryHandle source_his);
 70: long     CopyInd_sub(FunctionListHandle target_fnc,
 71:                   FunctionListHandle source_fnc,
 72:                          HistoryPtr his);
 73: long     NewIndividual(IndividualListHandle ind,
 74:                                 HistoryPtr his);
 75: long     NewHistory(HistoryHandle his);
 76: long     NewFunction(FunctionListHandle fnc,
 77:                                 HistoryPtr his);
 78: long     FinishAction(HistoryPtr    his,long x,long y);
 79: long     GetHistory(HistoryPtr,long);
 80:
 81: long     GoStraight(FunctionListPtr,long*,long*);
 82: long     TurnRight(FunctionListPtr,long*,long*);
 83: long     TurnLeft(FunctionListPtr,long*,long*);
 84: long     CheckHistory(FunctionListPtr,long*,long*);
 85:
 86: void     MovePosTo(long x,long y);
 87: void     LinePosTo(long x,long y);
 88: void     SetWindowPos(long pos);
 89: void     EraseWindowRect(void);
 90:
 91: void     main()
 92: {
 93: long     finished = false;
 94: IndividualListHandle     target;
 95:     screen(2,0,1,1);
 96:     console( 0,32,0,0 );
 97:     palet(1,rgb(31,31,31));
 98:     allmem();
 99:
100:     Individuals = (IndividualListPtr)0;
101:     Funcs[Fstraight] = (listed_func*)&GoStraight;
102:     Funcs[Fright] = (listed_func*)&TurnRight;
103:     Funcs[Fleft] = (listed_func*)&TurnLeft;
104:     Funcs[Fcheck] = (listed_func*)&CheckHistory;
105:
106:     target = GetLastIndividual();
107:     NewInd_his(target,&Histories);
108:     (*(**target).top_function).function =
109:                   (listed_func*)Funcs[Fright];
110:
111:     while( !finished && !kbhit() )
112:     {
113:         finished = Evolute();
114:     }
115:     return;
116:
117: }
118:
119: long     Evolute(void)
120: {
121: IndividualListPtr        target;
122: long     err = no_err,count = 0;
123:     target = Individuals;
124:     while( target != NULL && err == no_err ) {
125:         SetWindowPos(count);
126:         EraseWindowRect();
127:         (*(*target).theHistory).length = 0;
128:         ExecuteActions(target);
129:         target = (*target).next_individual;
130:         count ++;
131:     }
132:     target = Individuals;
133:     while( count > 0 && err == no_err ) {
134:         err |= Mutate(target);
135:         target = (*target).next_individual;
136:         count --;
137:     }
138:     return( err );
139: }
140:
141: void     ExecuteActions(IndividualListPtr ind)
142: {
143: FunctionListPtr target;
144: long     result = 0,x = 64,y = 64,c = 0;
145:     target = (*ind).top_function;
146:     (*(*target).theHistory).length = 0;
147:     while( result != state_stucked &&
148:            result != state_no_executes &&
149:            c < 4 ) {
150:         result = Execute(target,&x,&y);
151:         c++;
152:     }
153: }
154:
155: long     Execute(FunctionListPtr target,long* x,long* y)
156: {
157: long     result = 0;
158:     while( result != state_stucked ) {
159:         if( (*target).sub_function != NULL ) {
160:             result = Execute((*target).sub_function,x,y);
161:             if( result == state_stucked ) {
162:                 return(result);
163:             }
164:         }
165:         result =((*target).function)(target,x,y);
166:         switch( result ) {
167:           case state_false :
168:             if( (*target).next_function != NULL ) {
169:                 target = (*target).next_function;
170:                 if( (*target).next_function != NULL ) {
171:                     target = (*target).next_function;
172:                 }
173:             } else {
174:                 return(state_no_executes);
175:             }
176:             break;
177:           case state_true :
178:             if( (*target).next_function != NULL ) {
179:                 target = (*target).next_function;
180:             } else {
181:                 return(state_no_executes);
182:             }
183:             break;
184:           case state_stucked :
185:             return( result );
186:             break;
187:           default :
188:             if( (*target).next_function != NULL ) {
189:                 target = (*target).next_function;
190:             } else {
191:                 return(0);
192:             }
193:             break;
194:         }
195:     }
196:     return( result );
197: }
198:
199: long     Mutate(IndividualListPtr source)
200: {
201: IndividualListHandle     target_ind;
202: HistoryPtr       target_his;
203: long      err = no_err;
204:     target_ind = GetLastIndividual();
205:     if( total < 24 ) {
206:         CopyInd_his(target_ind,&target_his,
207:             &source,&(*source).theHistory);
208:         total ++;
209:         err = MutateFunc((**target_ind).top_function);
210:     }
211:     err = MutateFunc((*source).top_function);
212:     return( err );
213: }
214:
215: long     MutateFunc(FunctionListPtr target)
216: {
217: long      err = no_err,i,depth = 0;
218: FunctionListPtr tmp_func = target;
219:     while( (*tmp_func).next_function != NULL ) {
220:         tmp_func = (*tmp_func).next_function;
221:         depth++;
222:     }
223:     depth += depth*10/9;
224:     depth = (double)rand()/((long)RAND_MAX)*depth;
225:     while( depth > 0 && (*target).next_function != NULL ) {
226:         target = (*target).next_function;
227:         depth--;
228:     }
229:     if( (*target).sub_function != NULL ) {
230:         MutateFunc((*target).sub_function);
231:     } else {
232:         if( rand() < RAND_MAX/30 ) {
233:         err = NewFunction(&(*target).sub_function,
234:                           (*target).theHistory);
235:         if( err ) {
236:             return( err );
237:         }
238:         i = (double)rand()/((long)RAND_MAX+1)*Fmax;
239:         (*(FunctionListPtr)
240:             (*target).sub_function).function
241:                 = (listed_func*)Funcs[i];
242:         (*(FunctionListPtr)
243:             (*target).sub_function).par1
244:                 = (double)(rand())*50/
245:                     ((long)RAND_MAX+1);
246:         (*(FunctionListPtr)
247:             (*target).sub_function).par2
248:                 = (double)(rand())*Fmax/
249:                     ((long)RAND_MAX+1);
250:         return(0);
251:         }
252:     }
253:     if( (*target).next_function == NULL ) {
254:         err = NewFunction(&(*target).next_function,
255:                           (*target).theHistory);
256:         if( err ) {
257:             return( err );
258:         }
259:         target = (*target).next_function;
260:     }
261:     i = (double)rand()/((long)RAND_MAX+1)*Fmax;
262:     (*target).function = (listed_func*)Funcs[i];
263:     (*target).par1 =
264:             (double)(rand()-1)*100/(RAND_MAX+1)-50;
265:     (*target).par2 =
266:             (double)(rand()-1)*Fmax/(RAND_MAX+1);
267:     return(0);
268:
269: }
270:
271: IndividualListHandle     GetLastIndividual(void)
272: {
```


▶ビジネスショウに行ってきました。シャープの書院をインターネットのブラウザとして使っていました。「X68000用は開発していませんか？」と尋ねたところ，「それってなんですか？」といわれてしまいました。ぼくはそんなシャープが大好きです。

田辺　和広(25)東京都

```
273: IndividualListHandle    target;
274:     target = &Individuals;
275:     while( (*target) != NULL ) {
276:         target = &(**target).next_individual;
277:     }
278:     return( target );
279: }
280:
281: long    NewInd_his(IndividualListHandle ind,
282:                             HistoryHandle his)
283: {
284: long    err = no_err;
285:     err |= NewHistory(his);
286:     err |= NewIndividual(ind,*his);
287:     if( err == mem_err ) {
288:         return( err );
289:     }
290:     (**ind).theHistory = *his;
291:     return( no_err );
292: }
293:
294: long    CopyInd_his(IndividualListHandle target_ind,
295:                     HistoryHandle target_his,
296:                     IndividualListHandle source_ind,
297:                     HistoryHandle source_his)
298: {
299: long    err = no_err;
300:     err = NewInd_his(target_ind,target_his);
301:     if( err == mem_err ) {
302:         return( err );
303:     }
304:     memcpy(*target_his,*source_his,sizeof(History));
305:     memcpy(*target_ind,*source_ind,sizeof(IndividualList));
306:     (**target_ind).theHistory = *target_his;
307:     (**target_ind).next_individual = NULL;
308:     err = CopyInd_sub(&(**target_ind).top_function,
309:             &(**source_ind).top_function,
310:                     *target_his);
311:     return( err );
312: }
313:
314: long    CopyInd_sub(FunctionListHandle target_fnc,
315:                 FunctionListHandle source_fnc,
316:                             HistoryPtr his)
317: {
318: FunctionListPtr     source,target;
319: long            err;
320:     source = *source_fnc;
321:     target = *target_fnc;
322:     while( *target_fnc != NULL ) {
323:         err = NewFunction(target_fnc,his);
324:         memcpy((*target_fnc),(*source_fnc),
325:                     sizeof(FunctionList));
326:         (**target_fnc).theHistory = his;
327:         if( err == mem_err ) {
328:             return( err );
329:         }
330:         if( (**target_fnc).sub_function != NULL ) {
331:             err = CopyInd_sub(
332:                 &(**target_fnc).sub_function,
333:                 &(**source_fnc).sub_function,
334:                                 his);
335:             if( err == mem_err ) {
336:                 return( err );
337:             }
338:         }
339:         target_fnc = &(**target_fnc).next_function;
340:         source_fnc = &(**source_fnc).next_function;
341:     }
342:     return( no_err );
343: }
344:
345: long    NewIndividual(IndividualListHandle ind,
346:                             HistoryPtr his)
347: {
348:     *ind = (IndividualListPtr)calloc(1,
349:                 sizeof(IndividualList));
350:     if( *ind == 0 ) {
351:         return( mem_err );
352:     }
353:     NewFunction(&(**ind).top_function,his);
354:     (**ind).next_individual = NULL;
355:     (**ind).theHistory = his;
356:     return( no_err );
357: }
358:
359: long    NewHistory(HistoryHandle his)
360: {
361:     *his = (HistoryPtr)calloc(1,sizeof(History));
362:     if( *his == 0 ) {
363:         return( mem_err );
364:     }
365:     (**his).lengthPrev = 0;
366:     return( no_err );
367: }
368:
369: long    NewFunction(FunctionListHandle fnc,
370:                             HistoryPtr his)
371: {
372:     *fnc = (FunctionListPtr)calloc(1,
373:             sizeof(FunctionList));
374:     if( *fnc == 0 ) {
375:         return( mem_err );
376:     }
377:     (**fnc).next_function = NULL;
378:     (**fnc).theHistory = his;
379:     return( no_err );
380: }
381:
382: long    FinishAction(HistoryPtr his,long x,long y)
383: {
384:     (*his).position++;
385:     (*his).position %= 50;
386:     (*his).action[(*his).position] = (*his).direction;
387:     (*his).length++;
388:     return( 0 );
389: }
390:
391: long    GetHistory(HistoryPtr his,long offset)
392: {
393:     offset = (*his).position-offset + 50;
394:     offset %= 50;
395:     return( (*his).action[offset] );
396: }
397:
398: long    GoStraight(FunctionListPtr myself,long* x,long* y)
399: {
400: long    result = 0;
401:     MovePosTo((short)*x,(short)*y);
402:     *x += dir_x[(*(*myself).theHistory).direction];
403:     *y += dir_y[(*(*myself).theHistory).direction];
404:     result = FinishAction((*myself).theHistory,*x,*y);
405:     if( result ) {
406:         return(result);
407:     }
408:     LinePosTo((short)*x,(short)*y);
409:     return(go_straight);
410: }
411:
412: long    TurnRight(FunctionListPtr myself,long* x,long* y)
413: {
414: long    result = 0;
415:     MovePosTo((short)*x,(short)*y);
416:     (*(*myself).theHistory).direction++;
417:     (*(*myself).theHistory).direction %= 4;
418:     *x += dir_x[(*(*myself).theHistory).direction];
419:     *y += dir_y[(*(*myself).theHistory).direction];
420:     result = FinishAction((*myself).theHistory,*x,*y);
421:     if( result ) {
422:         return(result);
423:     }
424:     LinePosTo((short)*x,(short)*y);
425:     return(turn_right);
426: }
427:
428: long    TurnLeft(FunctionListPtr myself,long* x,long* y)
429: {
430: long    result = 0;
431:     MovePosTo((short)*x,(short)*y);
432:     (*(*myself).theHistory).direction += 3;
433:     (*(*myself).theHistory).direction %= 4;
434:     *x += dir_x[(*(*myself).theHistory).direction];
435:     *y += dir_y[(*(*myself).theHistory).direction];
436:     result =FinishAction((*myself).theHistory,*x,*y);
437:     if( result ) {
438:         return(result);
439:     }
440:     LinePosTo((short)*x,(short)*y);
441:     return(turn_left);
442: }
443:
444: long    CheckHistory(FunctionListPtr myself,
445:                         long* x,long* y)
446: {
447: long    result = state_true;
448:     if( (*myself).par1 > 0 ) {
449:         if( GetHistory((*myself).theHistory,1) ==
450:                         (*myself).par2 ) {
451:             return( state_true );
452:         }
453:         return( state_false );
454:     }
455:     if( GetHistory((*myself).theHistory,
456:             -(*myself).par1) ==
457:                 (*myself).par2 ) {
458:         return( state_false );
459:     }
460:     return( state_true );
461: }
462:
463: void    MovePosTo(long x,long y)
464: {
465:     wx = x+ox;
466:     wy = y+oy;
467: }
468:
469: void    LinePosTo(long x,long y)
470: {
471:     if( wx-ox > 128 || wx-ox < 0 ||
472:         wy-oy > 128 || wy-oy < 0 ||
473:             x > 128 || x < 0 ||
474:             y > 128 || y < 0 ) {
475:         return;
476:     }
477:     line(wx,wy,x+ox,y+oy,1,'NASI');
478:     wx = x+ox;
479:     wy = y+oy;
480: }
481:
482: void    SetWindowPos(long pos)
483: {
484:     ox = (pos % 6)*128;
485:     oy = (pos / 6)*128;
486: }
487:
488: void    EraseWindowRect(void)
489: {
490:     fill(ox,oy,ox+128,oy+128,0);
491: }
```

▶SX-WINDOWもよいのですが，僕にはグラフィック機能とスプライト機能を捨て去ることができません。というわけで，これからもHuman68k主体で使っていくことになりそうです。
浪越　孝宏(22)兵庫県

バックナンバー案内

BACK ISSUES

ここには1994年7月号から1995年6月号までをご紹介しました。現在1994年4～12月号，1995年4～6月号の在庫がございます。バックナンバーはお近くの書店にご注文ください。定期購読の申し込み方法は144ページを参照してください。

1994

7月号
特集　入門コンピュータミュージック

連載　響子 in CGわ～るど/ショートプロ/ゲーム作りのKNOW HOW
ローテク工作/システムX探偵事務所/マシン語プログラミング
DōGA CGアニメーション講座/ファイル共有の実験と実践

●特別付録　CGA入門キット「GENIE」
●実用講座　Photo CDでカードを作る
LIVE in '94　宇宙刑事ギャバン/究極戦隊ダダンダーン/スティング　他
THE SOFTOUCH　麻雀航海記/雀神クエスト/The World of　X68000II 他
全機種共通システム　シューティングゲーム作成講座(1)

8月号
特集　Graphic Movement

連載　響子 in CGわ～るど/ショートプロ/ハードコア3D
ローテク工作/ANOTHER CG WORLD/善バビ
DōGA CGアニメーション講座/石の言葉，言葉の夢

●新製品紹介　X-SIMM VI/Mu-1 GS
SX-WINDOW ver.3.1
LIVE in '94　PURE GREEN/Ridge racer(POWER REMIX)
THE SOFTOUCH　Mr.Do!/Mr.Do! vs UNICORNS/レッスルエンジェルス3
全機種共通システム　シューティングゲーム作成講座(2)

9月号
特集　SX-WINDOW環境セットアップ

連載　響子 in CGわ～るど/ショートプロ/ハードコア3D
ローテク工作/DōGA CGアニメーション講座/善バビ
システムX探偵事務所/ファイル共有の実験と実践

●新製品紹介　X68030 D'ash/MJ-700V2C
●新刊紹介　X680x0 TeX
LIVE in '94　LOVE IS ALL/HELL HOUND/踏切の通過音
THE SOFTOUCH　餓狼伝説SPECIAL
全機種共通システム　怪しいZ80の使い方(テクニック編)

10月号
特別企画　もみじ狩りPRO-68K

連載　響子in CGわ～るど/ショートプロ/ハードコア3D
TeX入門講座/ゲーム作りのKNOW HOW/善バビ
猫とコンピュータ/ファイル共有の実験と実践

●特別付録　もみじ狩りPRO-68K(5″2HD)
●新製品紹介　F-Card V5 for x68k
LIVE in '94　イース2/MSX用GRADIUS2/NATURE
THE SOFTOUCH　スーパーストII/スターラスター　他
全機種共通システム　怪しいZ80の使い方/ゲーム作成講座(3)

11月号
特集　STEP UP BASIC

連載　響子 in CGわ～るど/ショートプロ/ハードコア3D
TeX入門講座/DōGA CGアニメーション講座
システムX探偵事務所/ローテク工作/善バビ

●新製品紹介　BJC-400J/X680x0 Develop. & libcII
Free Software Selection Vol.2
LIVE in '94　ダーク・スペース/ENDLESS RAIN/レナのテーマ
THE SOFTOUCH　スーパーストII/餓狼伝説SPECIAL
全機種共通システム　B-GALETS2

12月号
特別企画　XL/Imageお試し版＋α

連載　響子in CGわ～るど/ショートプロ/ハードコア3D
ファイル共有の実験と実践/DōGA CGアニメーション講座
システムX探偵事務所/ローテク工作/TeX入門講座

●特別付録　XL/Imageお試し版＋α(5″2HD)
●新製品紹介　H.A.R.P/XDTP SX-68K
LIVE in '94　幻想即興曲/きまぐれ オレンジ☆ロード 他
THE SOFTOUCH　魔法大作戦/スーパーストII
全機種共通システム　シューティングゲーム作成講座(4)

1995

1月号(品切れ)
特集　割り切って使うCD-ROM

連載　響子 in CGわ～るど/ショートプロ/ハードコア3D
ファイル共有の実験と実践/DōGA CGアニメーション講座
システムX探偵事務所/ローテク工作/TeX入門講座

●CD-ROMドライブ紹介　CS-CD301X/CDS-E/SCD-200
●新製品紹介　X68000XVI用アクセラレータXellent30
LIVE in '95　ぷよぷよ/ジムノペディNO.1/PRIME
THE SOFTOUCH　バックランド/上海 万里の長城/魔法大作戦
餓狼伝説SP 特別編/スーパーストII 特別編

2月号(品切れ)
特集　MicroProcessingUnit

連載　響子 in CGわ～るど/ショートプロ/ハードコア3D
SX-BASIC公開デバッグ/DōGA CGアニメーション講座
システムX探偵事務所/SX-WINDOWによるDTP

●特別企画　最新ゲーム機を見る
●新製品紹介　Datacalc SX-68K/シャーペンワープロパック
●1994年度GAME OF THE YEARノミネート作品発表
LIVE in '95　サムライスピリッツ/AFTER SCHOOL/白鳥の湖
THE SOFTOUCH　スーパーストII 特別編

3月号(品切れ)
特集　SoundEffects

連載　響子 in　CGわ～るど/ショートプロ/ハードコア3D
システムX探偵事務所/ファイル共有の実験と実践
ピコピコエンジン活用講座/SX-WINDOWによるDTP

●SX-WINDOW用ユーティリティ　どっち.X
LIVE in '95　魔法のプリンセスミンキーモモ/別れの曲
ファイナルファンタジーII/宇宙戦艦ヤマト完結編
THE SOFTOUCH　ディグダグ/ディグダグII/VIEW POINT
全機種共通システム　S-OSシステムコールライブラリ

4月号
特集　Let's Play Wonderful GAME

連載　響子 in　CGわ～るど/ショートプロ/ハードコア3D
システムX探偵事務所/ファイル共有の実験と実践
DōGA CGアニメーション講座/ローテク工作

●1994年度GAME OF THE YEAR発表
●新製品紹介　TS-6BS1mkII/MJ-5000C/MATIER ver.2.1
LIVE in '95　天聖龍/ファイナルファンタジーVI/
ANOTHER DAY/ハートオブザマッドネス
全機種共通システム　S-OSねちねち入門(1)

5月号
特集　Realize Graphic

連載　響子 in CGわ～るど/ショートプロぱーてぃ
ローテク工作実験室/SX-BASIC公開デバッグ
システムX探偵事務所/ANOTHER CG WORLD

●特別付録　Oh!電脳倶楽部
●新製品紹介　フォント＆ロゴデザインツール
LIVE in '95　ドラゴンセイバー/ミッドナイトレジスタンス 他
THE SOFTOUCH　ボンバーマン ぱにっくボンバー
全機種共通システム　S-OSねちねち入門(2)

6月号
特集　Open the SX-WINDOW

連載　響子 in CGわ～るど/ハードコア3Dエクスタシー
DōGA CGアニメーション講座/ローテク工作実験室
システムX探偵事務所/ショートプロぱーてぃ

●特別企画　X68000周辺機器パワーアップ計画
●新製品紹介　Xellent30s/学研統合電子辞書 for SX-Window
●第6回アンケート分析大会
LIVE in '95　クリティカルポイント/THE SUMMER OF '68 他
全機種共通システム　S-OSねちねち入門(3)/BLOCK DOWN

愛読者プレゼント

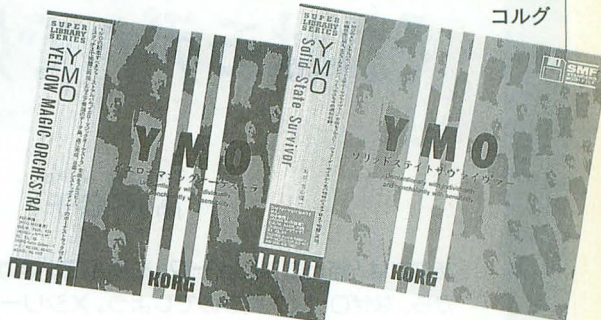
プレゼントの応募方法

とじ込みのアンケートハガキの該当項目をすべてご記入のうえ，希望するプレゼント番号をハガキ右下のスペースにひとつ記入してお申し込みください。締め切りは1995年7月18日の到着分までとします。当選者の発表は1995年9月号で行います。また，雑誌公正競争規約の定めにより，当選された方はこの号のほかの懸賞に当選できない場合がありますので，ご了承ください。

SUPER LIBRARY SERIES

A) YELLOW MAGIC ORCHESTRA/YMO
B) SOLID STATE SURVIVOR/YMO

3.5″2DD版　4,000円(税別)　各1名

コルグ

ミスタッチまで再現した高品位なスタンダードMIDIファイルデータ集です。制作者は「第6回CGAコンテスト」に「せっけんくん」で入賞した立岩氏。

対応音源：05R/W，X5DR，X3R，X2/3/5，AG-502/602/503/1002

スターモビール

X68000用　1名

5″2HD版　7,200円(税別)

M.N.M

重さの違う星を天秤にバランスよく乗せていく落ちものパズルゲーム。

スライス

1名

X68000用　5″2HD版　7,200円(税別)

M.N.M

同じタイプのブロックを重ねて消していく，コラムスタイプのパズルゲーム。古代祐三氏のBGMが面白い。

キーパー

1名

X68000用

5″2HD版　8,800円(税別)

サクセス　☎03(3791)2820

かわいいプクルが走り回る，ブロックアクションパズルゲーム。

A) THE World of X68000

X68000用　5″2HD版　4,800円(税別)

B) THE World of X68000 II

X68000用

5″2HD版　4,900円(税別)

各1名

電波新聞社　☎03(3445)6111

X68000芸術祭の入選作品を収録した書籍。本書のためにバージョンアップされたものもあるぞ。

5月号プレゼント当選者

1 A)PIECES OF WORK II （奈良県)寺元　正　B)ブレインボックス美術館　(大阪府)北浦新一　C)duplicity　(栃木県)狐塚一浩　D)HOPE　(北海道)宝福浩司　**2** A)シムシティ　(富山県)藤田和久　B)シムアース　(岐阜県)中野克巳　C)シムアント　(埼玉県)中村　健　**3** A)ダウンタウン熱血物語　(群馬県)須田圭介　(兵庫県)大畑佳史　B)ダウンタウン熱血物語　(大阪府)波多野雅章　(愛知県)久米和彦　C)ニュージーランドストーリー　(三重県)黒部浩孝　(千葉県)伴　武士　**4** 九十九電機オリジナルクリアファイル　(奈良県)菰田英和　(愛知県)村上哲也　(群馬県)久保田智久　(千葉県)大場育雄　他6名　(敬称略)

以上の方々が当選しました。商品は順次発送いたしますが，入荷状況などにより遅れる場合もあります。

猫とコンピュータ

それはモデムで始まった

Takazawa Kyoko
高沢 恭子

あなたは「猫とコンピュータ」を読んでいます。Oh!Xを買っているから。なぜOh!Xを買うのでしょう。Xシリーズをもっているから…
…ひとつの出来事がいろんなことにつながっているようです。

アキバを歩けば

5月の連休に帰京したときのこと。夫は東京宅の1200bpsのモデムを高速のものに更新しようと秋葉原に出かけた。

14400bpsまで可能な，OMRON（オムロン）のME1414Pという機種を購入したが，どうしたことか，新品のモデムははじめから故障の状態だった。

さっそく翌日，秋葉原の購入先の店にいき事情を話すと，なんの問題もなく，すぐに交換に応じてくれた。

その帰路，とある店先でPC-9801VXの新品が4台，積みかさねて縛られているのをみつけた。1台，9,800円で売るのだそうだ。

ほんのすこしのあいだに過去のマシンにされてしまったものの，会社に置けばじゅうぶん役にたつのは明らかだ。

1台買ってみようかと店の人に声をかけると，「これはいま，ひとりの人が全部買ったところです」という。それで縛ってあったのか。夫がちょっと残念に思っていると，店の人がいうには「あと1台だけ残っていますよ」。

奥から最後の1台というのが運ばれてきたのだが，こんどは夫が決断しかねて，やめますと答えた。だが，いったんことわって1分ほど歩いたとき，やはり買っておくべきだと判断を急転させて，店にひきかえしてみた。

「いま売れました」と店の人。そばに買い手の男性客が立っていた。ほんの2分ほどのあいだのことだった。

「それでね，なりゆきとはヘンなものだね。駅前の大道の売り屋で，こんなもの買っちゃったよ」と見せたのは，パソコンのキーボードくらいの大きさの，ブック型のビニールケースだった。

左右に開くと，水性のカラーペン，クレヨン，色鉛筆，パステル，水彩絵の具，文房具などがびっしり並んでいて，980円だそうだ。縁日のおみやげみたいだが，落書きくらいなら使えそうだ。

「モデムの交換に行ったら，VXに迷って最後は考えもしなかったものを買ってくるんだからオカシイね」という。

9,800円のあとに，980円が出てきたのもおもしろい。

かぎりないタネまき

すべてのできごとは，何かのつづきとして行われている。

なぜ東京宅のモデムを高速にしたかったかといえば，夫がこのごろインターネットを利用することが多くなったからだ。

三重の住まいではすでに高速のモデムを使っているので，東京でも同じようにしておきたいと思ったのだ。

そのためにグレードアップしたモデムをもとめに出かけ，そこで欠陥品を引き当ててしまった。そして交換のためにふたたび出かけることになった。

その外出がなければ，かつて30万円もしたVXの新品が9,800円で売られているのを見ることもなかったし，予想もしないオモチャのような画材のセットが持ち帰られることもなかった。

もとをたどれば，インターネットであり，さらにさかのぼれば限りない原因によってすべてが引き起こされている。

そんなことはあたりまえ。

きのうによって，たったさっきのできごとによって，私たちはいつも，つぎの行動を計画したり，余儀なくされたりしつづけている。

それは止められないし，止められてはならないはずである。

時を失う静かな朝

愛知県で市民劇団「演劇塾・火の鳥」を主宰する友人悦ちゃんから，この夏に自主公演を予定している『静かなる朝』という作品の台本が送られてきた。

アマチュアの人たちに，創作や表現，発表などの喜びを知ってもらい，同時におたがいを高めあう。地域の文化振興にも役だちたい。その活動のために，悦ちゃんはご主人とともに尽力している。

今回は戦後50周年にちなんでの公演ということだが，まず，作品選びがむずかしかったようだ。

130冊の台本を読んでみたが，よいと思うものがあっても，制作費の問題，それにともなって借りられる劇場の大きさ，演じる人たちの力量などと，さまざまな要因が優先されてしまう。けっきょく，採用した台本は30年も前のもので，原博さんという作者はすでに故人だった。

『静かなる朝』は，場面転換を行わない，シンプルで抽象的な1シーンの装置で演じられる。登場人物の衣装もすべて白一色である。

ときは1995年8月6日。この演劇が上演される当日である。

現代に設定された時間は，舞台そでに立った海軍服の男性のナレーションによって，あるふしぎな時間につれていかれる。

気持ちよく晴れた，静かな朝。ひとりの少女が妹をしたがえて，純白の洗濯ものを干している。目にしみるような青空。そこへ同い年と思われる4人の少女がつぎつぎに登場してくる。

それぞれが，明るい健康的な少女たちな

のだが，みんなが何かをやりかけていて，同じことをくりかえす動作から抜け出ることができなくなっている。

ひとりの少女はハンカチをさがしつづけ，ひとりは小さくなった靴になんとかして足を入れようとしつづける。別の少女は，自分がどこへいこうとしていたのか思い出せない。ひとりは，編んでも編んでもほどけてしまう長い髪を，左右かわりばんこに編みつづけている。

彼女たちの先生である女性も登場する。ところが，先生も弟にあてた手紙に，あとひとこと書き足そうとしたことを，どうしても思いだせないでいる。

みんな自分たちはどうかしてしまったらしいと，悲しく不安になる。そればかりか，いままで何をしていたのかも，わからなくなっているようなのである。

時々，時計を見るといつも「8時15分」なのだが，時計は止まっているのか，気のせいなのかもわからない。

先生は，自分が書いた手紙を読みかえしてみれば，いままでのことを知る手がかりになると気づいて，読みはじめる。

彼女は戦地の弟にあてて手紙をしたためていた。食糧をもとめに出かけたことや，防火訓練の話，学生たちの勤労奉仕や集団疎開の話など。その内容からみんな自分たちが，これからなにをしようとしていたのかを思いだしていく。

出征していく兄を見送りにいこうとしていたこと。大好きな靴をしまっておいたら小さくなってしまい，お米と交換すると父親にいわれたこと。さがしていたハンカチは，疎開していった同級生にもらった記念品だったこと。勤労奉仕に出かけるところだったこと。

みんなが，つぎにすることをさがしあてて，時計を見ると8時15分だった。登場人物が動きはじめようとしたところで，そのまま静止する。

舞台は一瞬の閃光，轟音，暗黒となる。

カゼがなおってラットに会う

昭和20年8月6日の午前8時15分に原爆は広島に投下された。

ふたたび，舞台そでに男性があらわれて語る。彼は先生の弟のようだ。彼女たちは50年たったいまも，時を失ったままどこかをさまよっているのではないか。姉が加えようとしていたひとことは「もう戦争はいやです」という言葉だったと。

悦ちゃんは，この台本があまりに古すぎて気にいらないのだというが，私はわかりやすい主張に共感できた。

原爆が落とされる直前の時間を止めて，つぎにつづくはずだった「時」を強調した。「失われた時」は「未来」であり，それを奪ったものへの怒りは，彼女たちの若さゆえにいっそう大きなものとして印象づけられる。

illustration：Kyoko Takazawa

現代の生活のなかでも，天災や人災でいわれなく時を奪われる人がいる。それにくらべたら，自分の時の流れのなかで，少々思いどおりにならないことと出会えるのは幸せといえる。

トオルは4月の末から，誕生以来はじめての高熱が出て，下がったと思うとまた発熱するような日がつづいた。

けっきょくは時季ハズレの流感だったようで，心配して行った検査結果にも異常はなかったのだが，そのために課題のレポートが未完成になった。

先生に許しをいただいて提出をのばすことができ，その猶予期間のおかげでレポートの内容は充実したものになった。

しかし，ゴールデンウィークは寝たままですごすことになり，そのためにサークル活動である吹奏楽団の，定期演奏会の練習を3回くらい欠席してしまった。

それからそれへと，「塞翁が馬」の現象がつづく。私もトオルといっしょに，のどカゼらしい症状になった。花粉症ではないかと夫がいいながら，花粉症になるプロセスの記事を私に見せてくれた。

日経新聞の記事の写しだったが，「風が吹けばオケ屋がもうかる」式のメカニズムという紹介ではじまっていた。

「花粉がとぶ→鼻や目の粘膜に付着する→血液中に花粉をつかまえる抗体が生じる→抗体が肥満細胞と呼ばれる細胞の表面にある受容体に結合する→肥満細胞が粘膜に集まる→肥満細胞の受容体に結合した抗体が花粉をつかまえる→肥満細胞が活性化し化学物質を出す→粘膜に炎症が生じる→花粉症になる」

なんとなく，新モデルのマシンが出たときのマニアの症状のようでもある。

トオルは健康をとりもどしたが，今回は高熱にコワサを感じたそうだ。「心理学研究法」の授業で行う「脳を見る」の実験に参加できるか否かは，熱が下がるか否かにかかっていた。

ただ，実験では脳を切除したネズミを使ったり，ホルマリンづけの人間の脳を机の上に取り出して，精密に描写するスケジュールになっていたので，熱は下がらなくてもいいかなと思ったそうだ。

めでたく実験日には登校できた。

脳の一部を取られ，助手の先生によって頭皮を乱暴に縫いあわされたネズミを見て，思わずダニエル・キイスの『アルジャーノンに花束を』を連想したそうだ。

「実験にはラットを使ったんだよ。マウスは小さいネズミ，大きいのがラットだよ」

そのあと「ボクが休めばラットの犠牲が減ったかなあ」と呟いた。

Another CGわーるど
vol 45
© KYOKO EGUCHI

さーてと

仕事するかー

ぐき

?
トントン

どうやら腰を痛めてしまったらしいのよー
とくに重いものをもったわけではないんだけど…

じつは、4年ほど前、ぎっくり腰で、1週間ばかり寝こんだことがあったのだ
まだ若いのに気のどくじゃのう〜
80すぎのおばあさんにまで同情された

で、今回は、早めに病院へ行ったところ…
68クリニック

お仕事は何ですか？
腰がいたいということですが
風邪は万病のもと
8

コンピュータで、デザインしたり、絵を描いたりしています…
9

1日にどれぐらいコンピュータに向っていますか？
ふむ
風邪は万病のもと
10

ここのところ忙しかったのでまる1日画面にはりついていることもありました
えーと
11

同じ姿勢を長く続けていることによる疲労ですよこれは…
うーむ現代病ですねぇ
風邪は万病のもと
12

腰の痛みがとれたら、軽い運動を毎日するように…
1時間ごとに10分は休み、コンピュータでの仕事は1日4〜5時間にして下さい
このままつづくとぎっくり腰になります…
13

ということだった…
守るのはちとムズカシーなー
14

みなさんも気をつけて下さいね…
15

PENGUIN INFORMATION CORNER
ペ・ン・ギ・ン・情・報・コ・ー・ナ・ー

NEW PRODUCTS

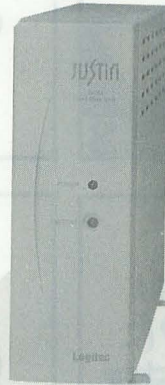
3.5インチ光磁気ディスク MOS-E230
メルコ

MOS-E230

メルコは3.5インチ光磁気ディスクドライブ「MOS-E230」を発売した。
同機の特徴は以下のとおり。
ディスク容量：128/230Mバイト
平均シーク速度：28ms
ディスク回転数：3,600rpm
バッファ容量：500Kバイト
インタフェイスはSCSI-2を採用。
価格は74,800円(税別)。
〈問い合わせ先〉
メルコ(株)　☎052(619)1827

ハードディスクドライブ SHD-BA340U/540U/1000U
ロジテック

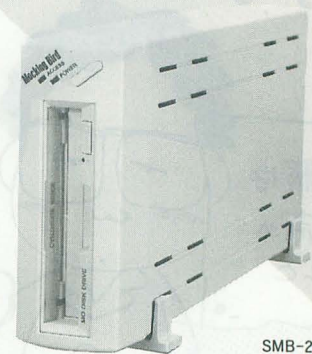
ロジテックはSCSI-2対応のハードディスクドライブ「SHD-BA340U」「SHD-BA540U」「SHD-BA1000U」3機種を発売した。
主な仕様は以下のとおり。
●SHD-BA340U
記憶容量：336Mバイト
平均シーク速度：12ms(リード)
15ms(ライト)
バッファ容量：256Kバイト
●SHD-BA540U
記憶容量：519Mバイト
平均シーク速度：10ms(リード)
12ms(ライト)
バッファ容量：256Kバイト

SHD-BA1000U

●SHD-BA1000U
記憶容量：1029Mバイト
平均シーク速度：11ms(リード)
11.5ms(ライト)
バッファ容量：256Kバイト
付属品はマニュアルのみ。
価格は「SHD-BA340U」が35,800円，「SHD-BA540U」が43,800円，「SHD-BA1000U」が77,800円(ともに税別)。
〈問い合わせ先〉
ロジテック(株)　☎03(3251)3271

3.5インチ光磁気ディスク LMO-450H
ロジテック

LMO-450H

ロジテックは3.5インチ光磁気ディスクドライブ「LMO-450H」を発売した。
同機の特徴は以下のとおり。
ディスク容量：128/230Mバイト
平均シーク速度：27ms
ディスク回転数：4,500rpm
バッファ容量：1Mバイト
インタフェイスはSCSI-2を採用。
価格は108,000円(税別)。
〈問い合わせ先〉
ロジテック(株)　☎03(3251)3271

3.5インチ光磁気ディスク SMB-230D
富士通OA

SMB-230D

富士通OAは3.5インチ光磁気ディスクドライブ“MockingBird-MO”「SMB-230D」を発売した。
同機の特徴は以下のとおり。
ディスク容量：128/230Mバイト
平均シーク速度：30ms
ディスク回転数：3,600rpm
バッファ容量：237Kバイト
インタフェイスはSCSI-2を採用し，スイッチで切り替え可能なアクティブターミネータを内蔵している。
価格は99,800円(税別)。
〈問い合わせ先〉
富士通OA(株)　☎03(5256)9403

液晶パッド“WiZ” PA-Z500
シャープ

シャープは液晶パッド“WiZ”「PA-Z500」を発売した。
同機はペン1本ですべての操作ができる新しい情報ツールである。表示部はFSTN液晶を採用し，横159×縦240ドットで13桁16行相当の表示が可能。文字入力は手書きで行うが，直接入力と変換入力の2通りから選べる。また，手書き文字登録により，くせ字のスムーズな認識ができる(1語当たり10画の場合，30語)。主な機能として，

PA-Z500

スケジューラ，約束リスト，アドレス帳，メモ帳，英和/和英/漢字辞典が用意されたほか，利用中にこれらの機能をいつでも呼び出せるクイックメモ機能がある。また，メモ帳機能には5タイプ23種類のフォームをあらかじめ内蔵している。ほかにも，光通信機能による，WiZ同士はもちろん，ザウルスや書院ワープロ，電子手帳などとの通信が可能。記憶容量は512Kバイトでユーザーエリアが約256Kバイト。大きさは84.5mm(幅)×134mm(奥行)×15.9mm(厚さ)で，重さが約159g(電池含む)。

価格は30,000円(税別)。

〈問い合わせ先〉

シャープ㈱☎06(621)1221，03(5261)7271

マッハジェットカラープリンタ
MJ-500C/MJ-800C/MJ-900C
セイコーエプソン

MJ-500C

セイコーエプソンはマッハジェットカラープリンタ「MJ-500C」「MJ-800C」「MJ-900C」の3機種を発売した。

「MJ－500C」はスーパーファインモードで解像度セミ720dpiの出力をサポートし，普通紙での印刷も可能。印字速度は漢字全角で83cpsを実現。

「MJ-800C」は「MJ-700V2C」の後継モデルで，解像度720dpiの出力をサポートし，モノクロ印刷では普通紙にも720dpiでの出力が可能。印字速度は漢字全角で133cpsを実現。また「MJ-500C」とともにハガキサイズからＡ４サイズの用紙に対応し，インタフェイスはパラレルとシリアルを１系統ずつ装備している。

「MJ-900C」はカラー/モノクロともに普通紙へ解像度720dpiでの出力をサポートしている。印字速度は「MJ-800C」と同じで，用紙サイズはハガキサイズからＡ３ノビサイズに対応。インタフェイスはシリアルとパラレルを１系統ずつと拡張スロット１基を装備している。

価格は「MJ-500C」が49,800円，「MJ-800C」が79,800円，「MJ-900C」が108,000円(それぞれ税別)。

〈問い合わせ先〉

マッハジェットカラーインフォメーション ☎0424(99)7111

ディスプレイジャック
MK-RGB21-15/S
満開製作所

MK-RGB21-15

満開製作所はディスプレイジャック「MK-RGB21-15」「MK-RGB21-15S」2機種を発売する。

同機はスーパーファミコンやPlayStationなどのRGB映像出力が21ピンマルチのみ対応の機器を，パソコン(X68000, PC-9801)などで使用されるRGB15ピン入力仕様の水平走査周波数15kHzに対応したディスプレイテレビに接続することを可能にする。

「MK-RGB21-15」はパソコンとRGBゲーム機の電源オン/オフにより入力ソースの自動切り替え機能を内蔵。また，音声出力用にヘッドフォン端子とLINE端子を装備。

「MK-RGB21-15」は自動切り替え機能はなく，音声出力端子もLINE端子のみ。

価格は「MK-RGB21-15」が19,000円，「MK-RGB21-15S」が4,900円(ともに予価)。

〈問い合わせ先〉

㈱ 満開製作所 ☎03(3354)9282

INFORMATION

第3回
DEP'95
ソニー・ミュージックエンタテインメント

ソニー・ミュージックエンタテインメントは第３回デジタル・エンタテインメント・プログラム「DEP'95」を実施する。

主な募集要項は以下のとおり。

●募集対象：エンタテインメントソフトのクリエイター(ゲームデザイナー，シナリオライター，CGアーティスト，プログラマーなど)

●応募資格：プロ/アマ，個人/団体，年齢，性別，国籍など一切問わない。

●応募部門：プロフェッショナルコースとアマチュアコースに分かれ，それぞれ作品部門と人物部門の２部門がある。

プロフェッショナルかアマチュアかは，現在マルチメディア関係の業務に携わっているかどうかで分かれる。

作品部門：マルチメディアタイトル制作のための企画および作品の募集。プラットフォームはなんでも可。原則として未発表の企画および作品で応募者が権利を有するもの。提出物は完成品でなくても可。

人物部門：マルチメディアタイトル制作のための人材を募集。審査の資料として過去の具体的な作品や仕事の実績を知ることができるもの。自己PRや小論文など，表現方法は自由。

●応募期間：1995年６月20日～９月30日

●審査方法：

一次審査(企画力審査)：規定応募用紙による書類と提出物による審査。

二次審査(人物審査)：審査員と個人面接を行う。

三次審査(最終審査)：有識者による審査。

●応募方法：応募希望者は規定応募用紙に記入し提出すること。応募用紙は電話やFAXなど下記の問い合わせ先に請求すれば，郵送される。

●賞：

・DEP Best Award：賞金100万円＋制作，発売プログラム

・各部門賞をコース別に選出。賞金(アマチュアコース20万円，プロフェッショナルコース50万円)＋育成，支援プログラム

〈問い合わせ先〉

㈱ソニー・ミュージックエンタテインメント ☎03(3475)6900, FAX03(3475)7358, NIFTY-Serve:RGE00613, www:http://www1.sony.co.jp/InfoPlaza/SME/Gallery

FILES Oh!X

このインデックスは，タイトル，注記――著者名，誌名，月号，ページで構成されています。夏がやってきました。まずは，海，それとも山？

一般

▶NEWS
シリコングラフィックス社がショールームを開設した話題やコダックが第２世代フォトCDを発表したニュースなど。――編集部，ASAHIパソコン，5・15号，8-11pp.

▶EDUCATION
コンピュータ教育開発センターが開発したゲーム「エネルギー環境教育～未来(みく)の選択～」を紹介する。――坂本伸之，ASAHIパソコン，5・15号，44-45pp.

▶98ユーザーのためのマッキントッシュ教室　13
アプリケーションの実行についてMacintoshとWindowsの違いについて考える。――荻窪圭，ASAHIパソコン，5・15号，82-85pp.

▶特集２　グループウェアって何だ
グループウェアに関わる企業の実情をレポートする。――編集部，ASAHIパソコン，5・15号，86-97pp.

▶GlobalInterface column
アメリカで開かれた第５回「コンピュータ，フリーダム＆プライバシー(CFP)」のレポート。――高間剛典，ASAHIパソコン，5・15号，108-109pp.

▶HEAD LINE NEWS
「'95東京おもちゃショー」のレポートや第８回CGAコンテストの募集など。――編集部，コンプティーク，６月号，８-９pp.

▶特集１　パソコンゲームができるまで!!
パソコンゲームの現状を調べたり，作成に携わる人の実情を取材したりしてパソコンゲームができるまでを探る。――編集部，コンプティーク，６月号，17-25pp.

▶特集２　めざせ！　ゲームの鉄人
ゲーム作りを職業にしたい人へのアドバイスとゲーム制作者養成学校の紹介。――編集部，コンプティーク，６月号，28-46pp.

▶'94コンプティークSOFT大賞
読者の投票により1994年度の優秀なゲームソフトを決める。大賞は日本ファルコムの「英雄伝説III」。――編集部，コンプティーク，６月号，108-113pp.

▶こだわりゲーム年代記
今回は日本ファルコムが発売したソフトからパソコンゲームの歴史を考察する。――与志田拓実，コンプティーク，６月号，134-135pp.

▶特集１　悪魔の裏技
1994年末～1995年春にかけて発売された新世代ゲーム機の裏技を紹介する。――編集部，電撃王，６月号，36-47pp.

▶特集２　GAME SCHOOL体験ツアー'95
４つのゲームスクールをメインで紹介し，ゲーム業界への最短ルートを考える。業界入門キーワード集つき。――編集部，電撃王，６月号，103-116pp.

▶第７回アマチュアCGAコンテスト結果発表！
コンテストの結果と入賞作のビデオの申し込み方法。付録CDにも一部作品の映像を収録。――編集部，マイコンBASIC Magazine，６月号，付録CD，44p.

▶Arcade Game Graffiti　第16回
1982年に登場したアーケードゲームを振り返る。今回は「ハンバーガー」「タイムパイロット」などを紹介する。――編集部，マイコンBASIC Magazine，６月号，142-145pp.

▶ゲーム考現学　第11回
ゲームのなかで扱われる素材について考える。今回は魔法について。――山田整＆桂令夫，マイコンBASIC Magazine，６月号，156-157pp.

▶特集１　サブノートマシンはこうして選ぶ
目的別に８機種のサブノートマシンを比較検討する。――編集部，ASAHIパソコン，6・1号，18-30pp.

▶98ユーザーのためのマッキントッシュ教室　14
WindowsとMacintoshのCD-ROMの扱いの違いを解説する。――荻窪圭，ASAHIパソコン，6・1号，106-109pp.

▶MultiMedia Watching　18
マルチメディアの著作権問題や各メディアで広がるデータ放送などについて紹介する。――奥野雅之，I/O，６月号，71-73pp.

▶手軽に使うイメージスキャナ
イメージスキャナの簡単な使用例と各社製品の紹介。――編集部，I/O，６月号，81-88pp.

▶インターネットアクセスガイド　１
１回目の今回はインターネットの概要やwww，アクセス方法などについて説明する。――森羅万象，I/O，６月号，89-90pp.

▶「掃除」で快適パソコンライフ
本体やマウスなどの市販キットを使った掃除方法を紹介する。――南雲徹，I/O，６月号，91-92pp.

▶インタ一ネット＆パソ通
ホームページの紹介やインターネットに関するＱ＆Ａ，用語集，パソコン通信のキーマンへのインタビューなど。――竹本隆ほか，I/O，６月号，93-106pp.

▶ターミネータの話
ターミネータの働きの解説と優れもののターミネータの紹介。――松枝知直，I/O，６月号，125p.

▶Desk Top Music入門　３
今回はMIDIの仕組みについて簡単に解説する。――てんてん，I/O，６月号，134-137pp.

▶特集１　モービルコンピューティング
携帯マシンを使ってのデータ交換，リモートアクセスなど。最新携帯マシンのレビューも盛りだくさん。――編集部，ASCII，６月号，265-291pp.

▶平成パソコンMONO選び
パソコン関連の小物を大紹介。キーボードやマウスからお掃除キットまで。ライターのお勧めグッズもあり。――編集部，ASCII，６月号，301-316pp.

▶Wozの魔法使い　第４回
今回はAppleIIのグラフィック機能を解説する。――柴田文彦，ASCII，６月号，363-365pp.

▶魅惑のニューテクノロジー
各社のチップセットに焦点を当て，関連するハードウェア技術を紹介する。――編集部，ASCII，６月号，370-375pp.

▶INTERNET膝栗毛　ROUTE５
インターネットに関する話題。ホームページの書き方や接続日記など。――編集部，ASCII，６月号，380-384pp.

▶特別企画　旅行者のためのパソコン情報
携帯マシン４機種の乗り物の中での使いごこちを検証する。――編集部，ASCII，６月号，415-419pp.

▶特集　メイキング　オブ　ぴゅーりほーCG
美しいCGを書くためのコツを手順別に解説する。――編集部，LOGIN，11号，139-155pp.

▶インターネットの心
初心者のためのインターネット接続講座や関連トピック紹介。またLOGINのホームページも紹介。――編集部，LOGIN，11号，188-191pp.

▶架想楽園へ行こう Ver.2.04
「2001年宇宙の旅」で特殊効果スーパーバイザとして参加したダグラス・トランブルの足跡と未来へのビジョンを探る。――中田宏之，LOGIN，11号，192-195pp.

▶くねくね科学探検隊　第20回
今回は瞑想とは何か？　その方法，目的などについて考える。――鹿野司，LOGIN，11号，208-211pp.

X1/turbo/Z

X1シリーズ

▶PURU・PURU・PURU
タイムを競うパズルゲーム。――青山正美，マイコンBASIC Magazine，６月号，100-101pp.

X68000

▶SOFT TOP 20
発売中のパソコンゲームソフトのトップ20。16位にX68000用の「ディグダグ/ディグダグ２」が登場。今月の赤丸チェックのコーナーにも登場。――編集部，コンプティーク，６月号，12-13pp.

▶SUPER SOFT INDEX
機種別の新作予定表。X68000用は「地球防衛 Miracle Force」など。――編集部，コンプティーク，６月号，117-118pp.

▶電撃新作予定表
新作の発売予定表。X68000用は「バラデューク」など。――編集部，電撃王，６月号，194p.

▶SHADOW
２人対戦ゲーム。止まることのできない自分を操作して，ダメージブロックで相手を倒す。――阿部啓一郎，マイコンBASIC Magazine，６月号，102-105pp.

▶レイフォース～エンディングミュージック～
ミュージックプログラム。NAGDRV2＋GS音源用。――REAL，マイコンBASIC Magazine，６月号，112-113pp.

▶SUPER SOFT Hot Information
X68000用は電波新聞社の「バラデューク」などを紹介。――編集部，マイコンBASIC Magazine，６月号，とじ込み付録12p.

▶ONLINE SOFTWARE INDEX
大手ネットにアップロードされたプログラムを紹介する。X68000用はSXでのシステムエラーを回避する「SXerror.X」とPhotoCDのローダ「PCD68k.X」。――編集部，ASCII，６月号，500-501pp.

▶GameReview
「ぱにっくボンバー」を３人のライターがレビューする。――編集部，LOGIN，11号，264p.

▶SX-WINDOWプログラミング　第20回
数回に分けて作成したファイルカッターを完成させる。――吉野智興，C MAGAZINE，６月号，130-135pp.

ポケコン

PC-E500

▶無無死
１～９の数字を並べて役を作るゲーム。２人用。――命かけます授業中の兄，マイコンBASIC Magazine，６月号，106p.

参考文献

I/O　工学社
ASAHIパソコン　朝日新聞社
ASCII　アスキー
コンプティーク　角川書店
C MAGAZINE　ソフトバンク
電撃王　主婦の友社
マイコンBASIC Magazine　電波新聞社
LOGIN　アスキー

QUESTION and ANSWER

Oh!X 質問箱

先日X68000ユーザーの友人がうちにやってきて私のマシンを使っていたのですが,

APATH

と打ち込んで,システムにコマンドがないといわれて驚いていました。

友人のマシンではAPATHというコマンドが有効なのに,うちのマシンではそのようなコマンドはありません。BINディレクトリにAPATH.Xというのはありませんから,内部コマンドだと思います。ちなみに,Human68kのバージョンは同じものを使っています。どうなっているのでしょうか。

東京都　田辺武司

田辺さんのおっしゃるように標準外部コマンドにAPATH.Xなどというものはありませんし,別に,

A>APATH

と打ち込んでもディスクは回りませんから,内部コマンドだと思ってしまうのも無理はありません。

結論をひと言でいえば,これはエイリアスです。標準状態のHISTORY.HISの中身をのぞいてみればよくわかると思います。

要するに,既存のコマンド指定の文字列をシステムが名前つきで疑似コマンド化したものなのです。

ということで,まず,システムを確認してください。HISTORY.Xを組み込んでいますか? これはHISTORY.Xの機能によって拡張されたものですので,HISTORY.Xを組み込んでいなければ使用できません。また,標準設定以外の組み込み方をしていると使えないかもしれません。そのあたりに注意して確認してみてください。

HISTORY.HISをよく見ると,APATH以外にもいくつか設定がなされているのがわかります。LPFFはキーボード操作(OPT.1+COPY)でできるので無意味な気もしますが,バッチファイルなどで使用するには便利なのでしょうか。いずれにせよ,エイリアスを使えば,こういったお決まりの操作を簡易コマンド化することができます。HISTORY.Xは非常に多機能なので一度マニュアルをじっくり読み直してみることをおすすめします。

質問です。AMIGAのことですが,よくグラフィックアーキテクチャが優秀と聞きますが,「ビット単位のDMAC」という表現がまったくわかりません。「VRAMがない」というのも画面状態を保存せずにどうやって画面を更新しているのでしょうか。またDOSマシンのWINDOWSアクセラレータによるメモリ→VRAM転送も謎です。VRAMのポートはSAMポート並みのアクセスが可能なんでしょうか。単純転送ならVRAMのアクセス速度で転送速度が決まってしまうのでCPU転送と速度は変わらないように思うのですが,どうなのでしょうか。

長崎県　加藤泰法

まずAMIGA関係ですが,AMIGAには他機種のVRAMに相当するものとして,ChipRAMという領域が用意されています。たとえば,AMIGA500にはChipRAMが512Kバイト用意されています。ところがAMIGA500に搭載されたRAMは全部で512Kバイト,要するにメインメモリとVRAMが同じ……といった感じになっているのです。

AMIGAのアプリケーションはたいてい全画面分の領域を占有しますが,そういったものを同時に複数起動して,あたかもそれぞれが1画面分のVRAMを持っているかのように重ねて表示することも可能でした。

画面として決まった領域を持たず,コントローラにコマンド列とデータを送って画面を構成しています。そういったコマンド列を蓄える領域がChipRAMなのです。

AMIGAにはメモリ付きのアクセラレータもたくさん出ていますが,DMAを使う領域が固定されているので,なにも気にせずローカルRAMを使用できます。これがX68000ならどうやってローカルRAMまでDMAを届かせるか非常に苦労するところです。

このChipRAMからデータを各部に転送するのがAGNUS/ALISといったDMACです。ビット単位のDMACとは,ビット単位で位置指定のできるという意味と思ってください。AMIGAのグラフィックデータはいわゆる水平型ですが,バイト境界やビットシフトなどを気にすることなく任意の位置に表示できるわけです。

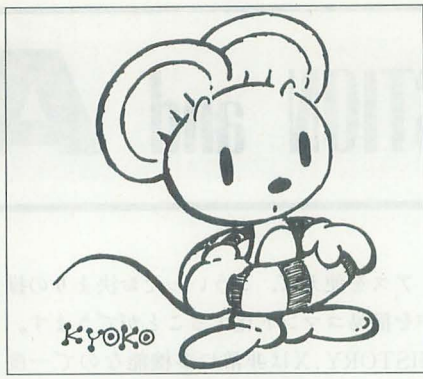
「VRAMがない」というのは,ChipRAMが汎用のDMA用データバッファであって,画像だけに限らず音声なども同様に処理されるからです。

次にWINDOWSアクセラレータの話ですが,ああいった製品はSAM(シリアルアクセスメモリ)ポートなどで絶えずメインメモリからデータを送っているわけではありません。チップで用意されている矩形転送というのはVRAM→VRAM転送です。メインメモリからのデータは共有RAMを用意するかCPU転送するしかありません。

(中野修一)

質問にお答えします

日ごろ疑問に思っていること,どんなことでも結構です。どんどんお便りください。難問,奇問,編集室が総力を挙げてお答えいたします。ただし,お寄せいただいているものの中には,マニュアルを読めばすぐに解答が得られるようなものも多々あります。最低限,マニュアルは熟読しておきましょう。質問はなるべく具体的に機種名,システム構成,必要なら図も入れてこと細かに書いてください。また,返信用切手同封の質問をよく受けますが,原則として,質問には本誌上でお答えすることになっていますのでご了承ください。なお,質問の内容について,直接問い合わせることもありますので電話番号も明記してください。

宛先:〒103　東京都中央区日本橋浜町3-42-3
ソフトバンク株式会社出版部
Oh!X編集部「Oh!X質問箱」係

FROM READERS TO THE EDITOR

やってきました日本の夏！ 開放的な海へ行くもよし，涼を求めて山へ行くもよし，暑さなんか吹っ飛ばしてめいいっぱい遊びましょう。日頃の運動不足がたたって夏バテなんかしないように，いまから体力作りにはげもうね。

◆私は，「地形」や「巻貝」などの自然界のものの形状を生成するプログラムを以前からほしいと思っていました。他機種にはこの手のCGソフトがあるので，いままでそれらのソフトを見ながら悔しい思いをしてきましたので，5月号の特集は実に嬉しいです。それから，EX-Systemですが，CD-ROMでもかまいません。早く出してほしいです！　本田　英雄(25)埼玉県

◆EX-Systemに期待しています。しかし，X68000でCD-ROMとは……おそらくマッピングデータが中心になってくるだろうと思うけど。Xellent30に対応しているのはいいかも（Xellent30はもっていないけど）。　白井　保弘(26)三重県

◆「ちゃだワ」でいっていた方もいらっしゃいましたが，初心者のための記事というものが最近のOh!Xでは極端に少なくなってきているような気がします。DOSの扱い方などの基本的なことを紹介する記事があってもいいのではないでしょうか。のほほんとパソコンを使うのもいいじゃないか，という感覚がOh!X誌上にもうちょっとほしいような気がします。
松本　祐一(25)青森県

◆X68000をこれから使ってみたいと思っているものです。情報の量がDOS/VやMacintoshに比べて少ないので，Oh!Xで初心者のための記事を掲載してもらえるとたいへん嬉しいのですが。
長谷川　透(23)石川県

◆先日，友人からX68000をもらい受け，初めてOh!Xを買って読んでみましたが，私にはだいぶ難しい内容でした。私のような初心者にもわかりやすくしてほしいと思います。あと，私がX68000を極めるまでがんばって続けてください。期待しています。　市原　一幸(19)東京都

ということでOh!Xに対する要望はアンケートハガキでガンガンお寄せください(なるべく具体的にね)。

◆5月号の付録ディスクの袋の中に「ウハウハ後払いスペシャル」の文字……なんかOh!Xと満開製作所「やるな！」って感じ（どんな感じ？）。しかし，満開製作所って北海道にもあるんだ。スゲエなあ。　島野　英男(20)東京都

◆「Oh!電脳倶楽部」は容量のせいか，月刊版よりパワーダウンしていましたね。これでは購読していない人に，あまり面白さが伝わってないような気がします。　浪越　孝宏(22)兵庫県

◆「Oh!電脳倶楽部」は思ったより怪しくなかったですね。もっと怪しくなると思っていたのに。
大野　隆士(23)沖縄県

◆「Oh!電脳倶楽部」がなかなか得した気分でした。なかでも「変酋長の雄叫び」の祝一平氏の言葉には考えさせられます。O!Xの恒例企画「言わせてくれなくちゃだワ」もやっぱりいいっすね。読者の生の考えがいろいろ見えて。あ，そうそう自画像を描き続けている酒井強氏も健在なのが嬉しいっす。それから「S-OSねちねち入門(2)」もよし。私にはわかるところとわからないところがあってワクワクします。最後にEX-Systemには期待大です。
三沢　弘之(23)神奈川県

◆5月号の付録ディスクは面白かったです。なかでもArtPadのドライバは助かりました。
竹川　貴彦(17)愛知県

◆「Oh!電脳倶楽部」のCDC.X，CDCSX.XはCD-ROMドライブを購入したばかりなので，タイミングがよかった。　増田　秀樹(28)東京都

さて，何人の読者が振込用紙を片手に郵便局へ走ったのでしょうか。

◆ようやくXellent30sが手に入った。取りつけには時間がかかったが，うまく動作してくれて嬉しい。今回初めてシールドを外したが，とても苦労した。クロックアップ改造をした人ってあんな作業をしていたんだ，と思うと驚きだ。数日後，MOドライブも買った。あとはメモリとCD-ROMドライブだ。　西尾　昌人(21)愛知県

◆Xellent30sがX68000 PRO対応でないことでブルー入っていましたが，「ちゃだワ」を読んでX68000ユーザーの考えがいまの私と同じだと感じ，少しだけほっとしました。
八亀　桂一(20)神奈川県

◆や，やっと出たMPUアクセラレータが！ Xellent30sの性能やいかに……。Oh!X 6月号の発売が楽しみ。　木植　幸男(23)東京都

Xellent30に続きXellent30sも評判は上々のようです。

◆前から思っていたことその1。X68000の背後にある電源スイッチは，いつもつけっぱなしにするものなんですか。　太田　志輝(18)北海道

基本的につけっぱなしでかまいません。あくまで，電気代をけちりたいなら，いちいち消せばいいだけの話です。が，定期的につけないとタイマ用のバッテリが上がって

◀岡村　直也　兵庫県
会話のテンポが合わなくてもラブラブ(死語)な2人。無事，バンジージャンプを一緒に飛んで(飛んでないって)2人の仲は進展した模様です。

◀千装　茂夫　埼玉県
なんだか，オモチャを買う約束をごまかし続ける母親みたい。なんとか永眠させないように復活させてあげてくださいね。

◀城間　裕樹　大阪府
根強いファンをもつコットン。コンシューマなどにも移植されているけど，やっぱりX68000版が一番よくできているよね。

▲牛島　真一　大阪府
僕もリブルラブルとメルヘンメイズは、ぜひとも確保したいなあ。でも、最近、コンシューマ関連でゲームを買いすぎて先立つものが……。

▲清家　亜紀　福岡県
大いなる野望に協力するため、シューティングゲーマー岩瀬さんのお隣に配置してみました。シューティング魂を吸い取ってうまくなるのだ！

▲岩瀬　貴代美　福岡県
カーバンクルはかあいいし、ハーピーの美声に悶絶しているアルルもいいぞ。ところでハガキのコメントがちょっとだけ投げやり。元気出してね。

しまうので注意してくださいね。

◆最近ようやくウィンドウなんか！(SX-WINDOWも含む)という感覚が消えました。なぜなのでしょうか。ところで，いつのまにPDやらZipとやらが世に出ていたんでしょうか。驚き……。
丸山　勝之(25)埼玉県
きっと大人になったんですよ。

◆僕が所属していたサークルにも新入部員がやってきた。「パソコン買おうと思ってるんですよ」というシロートな彼らに，僕はX680x0を勧められなかった。これからは，作るよりも使うマシンが求められることを思ったのだ。で，「ゲームも作りたいんです」という彼らに，僕は「FM TOWNSなんかいいんじゃない？」と答えてしまった。　奈良原　伸哉(22)福岡県
心のなかの葛藤が手に取るようにわかります。

◆同じクラスのH君がついにX68000ユーザーになった(中古のCZ-634CとCZ-614Dおよび新品の540Mバイトハードディスク)。約22万円したそうです。MSXでならした彼は，MIDIデータはあっても楽器がない，DōGAが2Mバイトのメモリでは思うように使えないなど，金欠病の影響で苦悩しています。まあ，彼より早く目をつけながらX68000で遊べない人がここにいますが。
佐久間　利浩(17)千葉県
確かにマシンを持っていないと遊べませんね。ここは友人という立場を最大限に利用して，H君と一緒にX68000を楽しんでみてはどうかな。

◆現在，PC-486SRを使いC言語でプログラムを組んでいます。ゲームを作っているのですが，スプライトがないため手をやいています。この前，X68000の「スーパーストリートファイターII」や「悪魔城ドラキュラ」を見てすごいと思い，X68030を買うためにお金を貯めています。
橋本　大輔(16)福島県
X68030を買ったからといってすべてが解決するものでもありません。お金を貯めながら，投げ出さずできることから解決するように努力してみては？

◆Drawing Pad(Slateじゃない)を中古で買ったのですが，「MATIER」で使えず3日ぐらい悩んでしまいました。Caltabは動くのに。とりあえず原因はわかったのですが，マニュアルにひと言「RSDRV.SYSが必要です」と記載してくれれば悩まずにすんだろう。　三枝　史浩(27)兵庫県
ということでマニュアルの改訂版にはひと言忘れずにね。サンワードさん。

◆私は，自分も知らない間に有田隆也先生の手先となって仕事をしていたようだ。今度，祝一平氏の過去を聞いてみたい。しかし，若く見えるがいったい何歳なのだろうか。
森　孝夫(23)愛知県
やはり謎は謎のまま残しておくのがいいかと思うのですが……。

◆大学4年生ということで研究室へ配属となり，必要上某国民機を購入することになった。友人などからあらかじめ聞いてはいたが，メモリ設定の複雑さには閉口している。しかし，X68000ではメインメモリ2Mバイトで，しかもハードディスクなしの状態で使用していたのに対し，メインメモリ13.6Mバイト，500Mバイトハードディスク，CD-ROMドライブ……いったいなにに使うのだろう。　山下　周大(21)岡山県
6畳のワンルームから4LDKの一軒家に引っ越した気分でしょうか。

◆皆さん，今年のゴールデンウィークはいかが過ごされたのでしょうか。私は三鷹の寮から鎌倉目指して自転車(MTB)をこいでいました。だいぶ大回りしたのに5時間かからなかったのにびっくり。勢い余って三浦半島の城ケ島まで行ってしまいました。帰りはヘトヘトだったので，愛車を袋に詰めて電車に乗って帰りました。さすがにゴールデンウィークの電車は，乗客数がいつもの半分以下しかいませんね。おかげでゆっくり座ることが(眠ることが？)できました。有意義な1日を過ごすことができてよかったです。　北本　信幸(22)東京都
今年のゴールデンウィークは「ジャンピングフラッシュ」やって，X-BASICでゲーム作って，パチンコ打っていたら終わってしまった。楽しかったからいいんだけど，ちょっとだけむなしいかな。

◆5月号を買って家に帰って本を見たらびっくり！　バッグに水が入っていて，本がフニャフニャになっていた。ディスクにも水がしたたって，よい男ならぬ，水もしたたるよいディスクに……。しようがないので次の日にまた買いました。このハガキは新しい5月号から切り取りました。　直井　崇仁(25)神奈川県
売り上げにご協力いただきありがとうございます。

◆自分自身のアクセルをめいっぱい踏んでみたい。でもいまは，突っ走るための道がない。
中島　民哉(24)埼玉県
よく，自分の進んだあとに道ができる，といいます。とりあえず目標を定めて突っ走りましょう。世の中なんとかなるものです。

◆アンケートハガキが斜めに綴じてあった。ちなみに定期購読の振替用紙はもっとひどくて，本体に綴じてある部分がちょっとしかありませんでした。時代とともにOh!Xまでも……？　ちょっと考えすぎかな。　中村　慶彦(16)山口県
それは立派な乱丁です。本屋さんにいって取り換えてもらいましょう。

◆どうでもいいことかもしれないが，氏名(フリガナ)がちと狭い。　西山　新志(24)福岡県
名前の横にフリガナを書けば，多少は余裕ができるかな。

◆以前，THE SENTINELにあったS-OSユーザーズクラブの「X1用不揮発外部メモリキット」が届きました。便利ですね。またこのような情報があったらぜひ紹介してください。
山中　雅彦(34)新潟県
ほいきた合点承知の助（なんだこりゃ）。

◆学校でIBMのノートパソコン「Think Pad」を買わされ，現在使っています。情報処理関係の授業だけでなく，一般科目でもレポート提出用に使うそうです。これからC言語も習うので，X68000にも応用していきます。
谷岡　学(18)山口県
がんばりましょう。マスターできればいい授業の暇潰しもできることですし(ちょっと違うか)。

◆「グラディウス」のコンティニュー方法がわかったので，さっそく試そうと6面までいったときESCキーと間違えBREAKキーを押してしまい，ゲームが終了してしまった。

北畠　駿(13)滋賀県

そうやって，人は間違いを繰り返しながら大人になっていくんだよ。

◆フリーターとしてブラブラしていたときには気がつかなかったけれど，仕事に追われる日々の合間に訪れる休日ってのは，本当にありがたいものですね。思えば昔はずいぶんともったいないことをしていたんだなあって(笑)。

堂領　輝昌(21)福岡県

この仕事をするようになってから休日が印刷所と写植屋が動かない日というふうに認識されてしまいました。へんなところに休日があると進行がきつくなるし……赤い曜日なんて嫌いだ～，と思わず夕日に向かって叫びたくなってしまいます。

◆仕事では，パソコンを使いたくないといいながら，仕事中にパソコンに向かっている自分。とはいっても実はパソコンで落書きをして遊んでいるだけなので，仕事ではありませんね。いや，仕事中のことだから一応仕事かな。う～ん，いったい私はなにをしているのか。

藤原　彰人(25)岡山県

遊んでいるんでしょ(みもふたもないか)。

◆高専から大学に編入したが，いろいろとつらいことが多い。やっぱり勉強しなけりゃあかんなあ。ところで，大学でインターネットへのアクセスやTeXを扱った講座があるようなので，受けてみようと思っています。

小海　昌伸(20)栃木県

そうそう，前向きな姿勢が大切。せっかく自分から望んで進路を決めたんですからがんばらなきゃ。

◆5月号の「STUDIO X」の安井百合江さんへ。遊び方がわからないということですが，それにはまずいろんな人に出会うことから始めるべきでしょう。異性でも同性でも，なんでも話せる友達を増やせば，いろんな考え方，価値観の異なった人たちに触れることができますし，いろいろなことに関心をもてるようになると思います。そこからいままで興味のなかったことに関心をもてるようになるでしょうし，なにか夢中になれることが見つかるでしょう。

北浦　暁光(21)東京都

ということですよ安井さん。

◆社会人になってから1カ月。健康的な生活を送っていたんですけど，ゴールデンウィークに実家へ帰ったら元の生活に……。元に戻るのは簡単ですね(直すのがたいへんそう)。

黒武者　健一(25)奈良県

すでにまっとうな生活とは無縁となってはや○年。もう社会復帰はできないかも……。

◆「言わせてくれなくちゃだワ」には結局出せなかった。イラストは下手だし，文章力ないし。修行せねば。それはそうと，現在私は学校で68000のアセンブラを習っています。すでに知っていることばかりで，ちょっといい気分です。

津村　忠蔵(20)佐賀県

基本的なことでも見落としがちなことは結構あります。授業は真面目に受けようね。

◆“幻想でも操作できる”を売りにしているコンピュータが幅をきかせていますが，ガキの頃から机にかじりつくってのは，やはりなにかおかしい。子供の頃はもっとプリミティブな道具を使ったほうが頭の体操になるし，しょせんパソコンはプラットフォームにすぎないのだから。学校でもコンピュータに慣れる教育はしても，なにかを創り出す教育はしていないので，日本の未来もそんなに明るいとは思えませんね。

久保田　忠弘(33)埼玉県

いろいろ問題が取り上げられている学校教育。学歴社会が完全に崩壊したらいったい日本はどうなるのでしょうか。

◆3月20日，サリン吸いました！ 血圧178/98で脈が92！ 目はベガだし(縮瞳してて本当にそっくり)足は痺れてたいへんでした。ほぼ1カ月後の4月19日は，地元横浜駅で謎の毒ガス発生……なんとも物騒な世の中ですが，私は今日も平気なフリして日比谷線で通勤するしかありません。気をつけてもしようがないものはもうあきらめていますから……せいぜい自分の悪運を信じるしか自衛の道はないのだろうと思っています。皆さんも気をつけてね。

鼻節　浩夫(33)神奈川県

……この件に関しては，もはやなにが正しくてなにが間違っているのかよくわからない状況になっています。信じられるのは自分自身だけになってしまいそうです。

◆「第7回アマチュアCGAコンテスト」のビデオが届きました。今年は上映会場へ行けなかったため，早めに申し込んだので，かなり早い回の発送に間に合ったようです。ビデオの感想は，なるほど今年は密度が濃いですねえ。感動ものです。解説が別なのも見やすくてGOOD! ところで森山さんにひと言「こんなんできるのはあんただけやがな」。

中村　哲也(26)東京都

X68000の可能性を見せてくれた森山さんはすごいですよね。

◆いま「MASTER OF MONSTERS II」にはまっています。古いゲームですが，TAKERUで2,500円という値段につられて買いました。が，これが見事に当たり。RPGのモンスターを操って進化させながら敵を倒す。おお，なんてやりがいのあるゲームなんだ。まさに理想のシミュレーションゲームだったのです(もともとRPGもシミュレーションも好き)。シナリオ集は出ないのでしょうか。このゲームが多人数でプレイできればまた違ったゲーム感になるでしょう。ところで，近所の大型電化店でX68000の古いソフトがオール2,000円で売っていました。つい，「T&T」と「出たな!! ツインビー」を買ってしまったのですが，安くなるのは嬉しくもあり，悲しくもありますね。ああ，X68000の未来は……。

地野　勝実(22)石川県

確かに叩き売り状態ですから気にもなりますよねえ。

◆質問です。現在，日本で最も多く使われていると思われる「石井明朝」ですが，これを「書家万流」でフォントとして作ったとします。で，このフォントはフリーデータとなりえるのでしょうか。字体にも著作権があるのでしょうか。

遠藤　勝博(25)宮城県

あります。

◆4月にXellent30sを装着した。通常使用でのパワー不足も感じられず，10MHzによるショックから解放され，なかなか気持ちがいい。REND.Xに関しては，強力なパワーをいかんなく発揮している。個人的にはおまけで入っていたX68000用の起動画面にシビレルものを感じます。しかし，もう胸を張ってX68000をお勧めできないと

◀武田　正道　兵庫県
運転免許証ぐらいは暇な学生時代に取っときゃよかったかなあ。バイトにゲーセン、パチンコにすべての時間を費やしたことをちょっとだけ後悔。

◀森本　真　愛知県
緻密で完成度の高いものもいいけど、大胆なライン、力強い絵柄がとてもかっこいいイラストですね。

◀占部　哲彦　広島県
ふっ、ネタなんていくらでも……季節、変な人間、宇宙人と愛(?)などなど。STUDIO Xではあまりにも危ないネタでないかぎり大丈夫。

▲奈良原 伸哉 福岡県

先の道がまったく見えなくても，もうここまできたら突き進むしかありません。親愛なる読者がいるかぎりがんばりまっせ。

▲藤沢 実 東京都

なにやら身辺があわただしいようですが，担当は無責任なエールを送るぞ。人生真面目にやってりゃいいことあるさ。がんばれよ。

ころまできていると思う。

森本 真(20)愛知県

森本さんのような元気なユーザーがいるかぎり，まだまだがんばれるでしょう。

◆SEGA SATURNが100万台まできた。それにしても「セガール」に「アンソニー」とは。

進戸 健太郎(18)兵庫県

土星人の次は猿……。まあ，それはそれとして，食事を忘れるほど面白いゲームがあるよ，といいたいのでしょうが，我々スタッフの間では，結局あのCMはセガールのしつけがなっていないだけという見解に落ち着いています。

◆最近思ったことをいくつか。“元祖飛びゲー”は「ジオグラフシール」ではないのか。「セガール」という名前は本当に存在するのか(猿の名前だからいいのか)。「ケンちゃんラーメン」はいつまで新発売なのか。「ケンちゃん～」がいちばん気になる。

三浦 貴至(23)埼玉県

「ケンちゃんラーメン」は確かによくわからない部分が多いですね。謎が解明される日はくるのでしょうか。

◆夢を見た。そこは本屋で「Oh!○○」などが並んでいるところに，白黒でOh!Xサイズの本がある。「なんだこれは」と手に取ってみると「緊急速報!! NewXのすべて」というタイトルと，潰れまくっているコピーの写真のようなNewXの姿がある表紙の本だった。なぜか，すべてのページがコピーで作ってあり，それがホチキスでまとめてあった。「特別定価550円」と書いてあり，発行は「Oh!X編集長」だった。そこで目が覚めてしまい，中身は読めなかったのだが……も，もう一度見たい。

片倉 純也(20)宮城県

今度はちゃんと中身を読んでから目を覚ますようにしましょうね。もちろんレポートもよろしく。

ぼくらの掲示板

●掲載ご希望の方は，官製ハガキに項目(売る・買う・氏名・年齢・連絡方法……)を明記してお申し込みください。
●ソフトの売買，交換については，いっさい掲載できません。
●取り引きについては当編集部では責任を負いかねます。
●応募者多数の場合，掲載できない場合もあります。
●紹介を希望されるサークルは必ず会誌の見本を送ってください。

売ります

★ツクモオリジナルSCSI&RAMボード「TS6BSI-mkII」(完動，美品，箱，説明書あり)＋8MバイトSIMMメモリ(実装ずみ)＋SCSIハーフ・ハーフケーブルをセットで40,000円(送料込)で売ります。連絡は往復ハガキでお願いします。〒164 東京都中野区本町3-9-8信沢マンション302 橋本 和典(28)

★X68000 CompactXVI用2MバイトRAMボード「CZ-6BE2D」(コプロMC68882つき)を25,000円(送料込)で売ります。箱，付属品などすべてあり，完動品です。連絡は往復ハガキで。〒176 東京都練馬区向山2-22-31-103 形部 聖一(35)

★X68000用ビデオボード「CZ-6BV1」を10,000円，X68000 XVI用メモリボード「CZ-6B2EA」を20,000円，X68000 XVI用メモリモジュール「CZ-6B2EB」を20,000円でそれぞれ売ります。連絡は往復ハガキでお願いします。〒520-05 滋賀県滋賀郡志賀町小野朝日1-4-9 倉谷 圭

★X68000 XVI用メモリ「CZ-6BE2A」＋「CZ-6BE2B」を合わせて30,000円で売ります。また，イメージスキャナ「CZ-8NS1」を10,000円，サイバースティック「CZ-8NJ2」を5,000円，カラーインクジェットプリンタ「IO-735X(グレー)」を15,000円，カラーイメージユニット「CZ-6VT1」を10,000円で売ります。箱はありませんが，ケーブル，説明書つきです。連絡は官製ハガキでお願いします。ただし，取りにこられる方に限ります。〒565 大阪府豊中市上新田4-8-C-509 市川 健一(25)

★24ドット熱転写カラー漢字プリンタ「CZ-8PC3」を15,000円で売ります。完動品，説明書ありですが箱はありません。連絡は往復ハガキでお願いします。〒460 愛知県名古屋市中区富士見町4-7上前津サンハイツ305 林本 一成

★ローランドのMIDI音源モジュール「CM-64」＋「SC-155」＋シャープのMIDIボード「CZ-6BM1」をセットで80,000円くらいで売ります。バラ売りも可です。あと，48ドット熱転写カラープリンタ「CZ-8PC5」をリボン(黒，カラー1本ずつ)をつけて15,000円くらいで売ります。すべて，箱，説明書，付属品あります。連絡は往復ハガキでお願いします。〒341 埼玉県三郷市早稲田5-22-22 岩田 勝博(30)

買います

★S端子変換ユニット「XAV-1s」を4,500円，スキャンコンバータ「XVGA-1s」を15,500円，ローランドMIDI音源モジュール「MT-32」を10,500円，X68000CompactXVI用2MバイトRAMボード「CZ-6BE2D」を10,000円で買います。箱，説明書はなくてもかまいません。連絡は往復ハガキでお願いします。〒395 長野県飯田市上郷別府2674-3 斎藤 雄慈(17)

★カラーイメージユニット「CZ-6VT1」か「CZ-6VT1-BK」を送料込み40,000円で買います。説明書と付属品があれば箱はなくてもかまいません。連絡は官製ハガキでお願いします。〒739 広島県東広島市西条町田口2799-7 コーポIZ302号 藤井 宣匡

★21インチディスプレイテレビ「CU-21HD」を50,000円，アイワ以外の14,400bpsのモデムを10,000円で買います(ともに送料込)。連絡は往復ハガキでお願いします。〒061-32 北海道石狩郡石狩町花川南5上3丁目275 永井 秀和(20)

バックナンバー

★Oh!X1988年9月号を2,000円で買います。送料は当方で負担します。連絡は往復ハガキでお願いします。〒319-11 茨城県那珂郡東海村緑ヶ丘団地10-16 牧野 豊(24)

編集室から from E・D・I・T・O・R

DRIVE ON

このコーナーでは，本誌年間モニタの方々のご意見を紹介しています。今月は5月号の内容に関するレポートです。

●5月号の特集により，X68000のグラフィックによる表現がどの程度できるかわかったように思います。「XL/Image」の能力も実感できました。そして「巻き貝を作る」には驚かされました。というのは，高校(大学だったかな?)の数学の時間に頭を痛めたベクトルと三角関数を使って作図するとは予想もつかなかったのです。しかも，法線ベクトルや仮想曲線，レンダリングなどの複雑怪奇な作業をしたうえでの完成です(その言葉の意味はいまもよくわかっていませんが)。

コンピュータグラフィックというのは結構複雑な処理をしていると知っていたつもりでしたが，これほど複雑で高度な処理をしているとは思いもしませんでした。

壁谷　善嗣(35)　X68000 EXPERT, PC-9821As, PC-9801NS/E　宮城県

●5月号の特集ですが，「DōGA CGAシステム」と「XL/Image」をもっていないとダメというのは……。ちょっとまとまりのない感じでした。自動生成ならそれだけに絞っていろいろやったほうがよかったかもしれません。

そんななかで，「貝」がすごかったですね。いわれてみれば，確かに厚みがないし，ちょっとカクカクしてるけど最初はぜんぜん気づきませんでした。取り込み画像かと思っていました。こういうことが簡単にできるとなると，CGの可能性を再認識させられます。

石田　伯仁(22)　X68030, MZ-731, PC-8801mkⅡMR, PC-E200　神奈川県

●やってきました恒例の「言わせてくれなくちゃだワ」。相変わらず熱いですねぇ。いろいろと意見を述べるのは大変よいことだと思います。そのなかでも，やはりシャープさんへの要望が多いようですね。周りのパソコンやゲーム機がどんどんパワーアップするなかで，X68000シリーズだけが取り残されていくような気がしているのでしょう。

しかし，X68000でないと納得できない人々がたくさんいるから，このように次期Xに期待の声が大きく上がっているのでしょう。

とりあえず，私自身は半分諦めていますが，いまのX68000でもなんら不安などありません。皆さんもX68000シリーズとは本来どうあるべきか考えてみてはいかがでしょうか?

大上　幸宏(22)　X68000 PROⅡ　鹿児島県

●やはり，読者のほとんどがNEW Xの登場を心待ちにしているのでしょう。肝心のシャープさんのほうはそろそろ動いてくるのでしょうか?　一度X68000に心奪われてしまった人は，きっと心のどこかでX68000に初めて出会ったときの感動を期待しているのでしょうが，現実が厳しいのも明らかです。あまり過剰な期待をするのも考えものです。過去にとらわれ続け，新しい道を見落とさないことも大切だと思います。なんにせよ，できる限り我々もがんばるしかないでしょう。

小林　佳徳(21)　X68000 XVI　新潟県

●「フォント＆ロゴデザインツール書家万流」の紹介記事はわかりやすくて参考になりましたが，付属するらしい半角フォントがどんなものか，写真か印刷例で示してほしかったと思います。また，実際に作ったフォントの例もあってしかるべきだったのではないでしょうか。

矢野　啓介(21)　X68000 XVI　北海道

●「(で)のショートプロぱーてぃ」について意見があります。私の考えではこのコーナーでのプログラムの紹介は扱いが低くていまいちに感じています。たぶん，D.J.方式の紹介であることが裏目に出ているのだと思います。特にゲームの場合，その作者やゲーム独自の世界を作り出しにくくなっているのです。現実感がありすぎるのです。ですから，ゲームプログラムの投稿については「マイコンBASIC Magazine」そのままのようなページを2～3ページ作るといいのではないでしょうか。そうすれば，独自の世界に没入できるし，ゲームの紹介の仕方や与えるヒントまでも思いどおりにできるので，ゲームに向いていると思います。

鈴木　朝夫(21)　X68000, MZ-1500, X1 turboZ, PC-9801RA, PC-88VA2, PC-6601SR, FM-77AV40SX, MSXturboR, ZX-81　神奈川県

●「第7回アマチュアCGAコンテスト」ですけど，ビデオを買いました。個人的には一昨年のほうが笑える作品が多かったような気がします。しかし，クオリティは高いですね。全体的にSEが少ないのが少々気になりましたが，音を集めるのと，映像と同期するのがなかなか難しいんでしょうね。

奥田　直也(22)　X68000 ACE-HD, X68000 SUPER, X68030, MSX2, PC-E560　神奈川県

●「知能機械概論」でコンピュータ上の仮想生命の進化モデルの話がありましたが，これは前にやった遺伝子複写のようなものだけかと思っていました。5月号で紹介されたような生態系のモデルによるものもあるんですね。ちょっと思いついたことがあるので，生態系モデルのプログラムを組んでみようと思ったら……記事の後半でそのプログラムについて説明がありました。

私がいま考えているものだと創発度はあまり高くならないような気がしますが，作成目的も記事中のものとは少し違うし，普通は完成したプログラムが期待どおりに動いたことに喜ぶものですからよしとしましょう……って，まずプログラムを組まなくちゃね。

弦元　達也(24)　X68000 ACE-HD　香川県

ごめんなさいのコーナー

6月号　SOFTWARE INFORMATION

P.16　電波新聞社の「バラデューク」の価格が間違っていました。正しくは5,300円です。関係者および，読者の方々に大変ご迷惑をおかけしました。お詫びいたします。

6月号　大容量ハードディスク導入の手引き

P.67　1段目に出ている内蔵SCSIケーブルの価格が間違っていました。正しくは流通コード007 512 0302:4,000円です。関係者および，読者の方々に大変ご迷惑をおかけしました。お詫びいたします。

6月号　Xellent30s

P.74　「Xellent30s」の価格が間違っていました。正しくは54,800円です。関係者および，読者の方々に大変ご迷惑をおかけしました。お詫びいたします。

バグに関するお問い合わせは
☎03(5642)8182(直通)
月～金曜日16：00～18：00

お問い合わせは原則として，本誌のバグ情報のみに限らせていただきます。入力法，操作法などはマニュアルをよくお読みください。

また，よくアドベンチャーゲームの解答を求めるお電話をいただきますが，本誌ではいっさいお答えできません。ご了承ください。

Optimize された美しいコード

▶「プログラムなんて思いどおりに動けばいいんだ」という人がいます。もちろん，動かないプログラムは意味がありません。でも，それなりの速度が要求されることはよくあることです。「自分が作ったプログラムは遅くて……」と感じる人もたくさんいることでしょう。それを解決するためにはいくつかの方法があります。

今回の特集ではそんな方法のひとつ「最適化」を取り上げてみました。もちろん，プログラムは速く動けばいいというものではありません。メンテナンスの関係で視認性のよいプログラムを要求されることもあるでしょう。それに，最適化といってもプログラムの速度を上げるだけが目的ではありません。ただ，完成したプログラムを自分の技術で少しでも速く動かすということは，そのプログラム，マシンへの愛ともいえるのではないでしょうか。

さあ，皆さんも自分のプログラムを最適化してみませんか。

▶さて，5月号で予告しました，第11期愛読者年間モニタ当選者の発表ですが，応募者がまだまだ少ないようですので募集を継続します。我こそはという方は，住所，氏名，年齢，職業(学年)，使用機種を明記のうえ，本紙へのご意見をレポート用紙2枚程度にまとめたものを，

Oh!X編集部「愛読者年間モニタ」係

まで郵送してください。締め切りは7月18日(必着)とします。なお，第10期愛読者年間モニタの方には新しいモニタの方が決まるまで継続でお願いしたいと思いますので，よろしくお願いします。

▶また今年も夏を迎えました。そこで，皆さんの暑さを吹き飛ばすようなさわやかな暑中見舞いのカラーイラストを待っています。10月号で紹介する予定です。

▶「X68000マシン語プログラミング」「石の言葉，言葉の夢」は今月も著者多忙のため，残念ながらお休みです。

投稿応募要領

●原稿には，住所・氏名・年齢・職業・連絡先電話番号・機種・使用言語・必要な周辺機器・マイコン歴を明記してください。

●プログラムを投稿される方は，詳しい内容の説明，利用法，できればフローチャート，変数表，メモリマップ（マシン語の場合）に，参考文献を明記し，プログラムをセーブしたフロッピーディスクを添えてお送りください。また，掲載にあたっては，編集上の都合により加筆修正させていただくことがありますのでご了承ください。

●ハードの製作などを投稿される方は，詳しい内容の説明のほかに回路図，部品表，できれば実体配線図も添えてください。編集室で検討のうえ，製作したハードが必要な場合はご連絡いたします。

●投稿者のモラルとして，他誌との二重投稿，他機種用プログラムを単に移植したものは固くお断りいたします。

あて先

〒103 東京都中央区日本橋浜町3-42-3

ソフトバンク出版部

Oh!X「㋢㋷㋮㊔」係

SHIFT・BREAK

▶長期間封印しておいた「Murder Club-DX」のディスクが3枚ともカビに侵されて死んでしまった。1時間かけてタオルと水で丹念に磁性面のカビをこすった。驚いたことにドライブがディスクを認識した。時々リードエラーが出るが，指先でディスクへひたすらバイブレーションを与え続けると読み込む。ああ，いままでの人生で最高に幸せ！　(H)

▶偶然にも友達のO君とM君がカナダに移住した。ちなみにO君とM君はお互いまったく面識はない。2人とも自分自身と自分の可能性をもう一度見つめ直したいというのだ。2度の歓送会で2人を送り出したあと，小さな机の上で同じことをしようとしている自分に気がついた。長い戦いになると思うけど，きっとできるはずだよ……お互いにね？　(哲)

▶12Mバイトは狭すぎる。最低64Mバイトはほしい。画像をやるにも，音楽やるにも足らん。努力すればメモリ喰わんアルゴリズムも組めるけどさ。もうそんなところで頭使いたくないし(軟弱)。SGIのGWSがほしいなあ。ところで，シャープが2月頃出したDOS/Vノート，誰か買った？　液晶のシャープ，映りはどう？　(そろそろ旅立ち先を考えたい瀧)

▶引っ越しをした。場所は，葛飾区の柴又にある帝釈天のすぐ真横。演歌で有名になった「矢切の渡し」へは，部屋から2分ほどで行ける。朝夕6時と正午には，帝釈天の鐘の音が聞こえてくる。静かで，なにより風情があるところが気に入っている。欲をいわせてもらえれば，休日になると大挙して押しかける観光客がいなくなればいいのだが……。　(ats)

▶GWに読売ランドでバンジージャンプしてきた。今年4月にできたばかりだ。わざわざこんなところでバンジージャンプする酔狂な人間はそうはいないので，待つこともなかった。認定証をもらったが，スタンプを押すところが6カ所ある。6回飛ぶとなにかあるんだろうか？　ひょっとして本場ニューカレドニア(？)に御招待だったりして。　(I.K)

▶おいらもヂャンヂャンバリバリ買い物してまふ。個人輸入でモデム，洋服，BB弾にコンタクト。こうなりゃ食料も調達して，目指せ海外調達率100%(輸入に2週間……おいおい餓死するぞ)。しかし，調子に乗って転売目当てで買ったDX4が壊れていたのは痛かった，とほほ。と，これが6月号のおちだったりして。　(このネタ元祖のI.K氏に。すまん(で))

▶移植もののレビューの常套手段で，直前に本物を遊び倒して原稿を書くことが多い。古いのや珍品は知人を頼るのだが，厖大な在庫からモノを捜すだけで果てたり，借りたはいいが配線表がないなどの苦労が絶えない。しかしレビューのためだというと，気軽に遊ばせてくれるので，今度騙して遊びまくってみよう。あとが怖いけどね，うんうん。　(八)

▶電脳倶楽部でも話題になったゲーム「海腹川背」はプログラマをも熱くさせる入魂の作品だ。ゴム紐の物理的挙動を余すところなく再現した快挙に拍手を送りたい。売れ筋の超A級大作には心惹かれない私だが，これのためになら本体を買ってもいい。まれにでもこうした秀作が出るのがスーファミの層の厚さか。　(でも難しくて挫折しそうなA.T.)

▶電話代を払いに行った。確かテレフォンカードで払えると聞いていたが，だめだという。支払いに当てられるのは通話料金だけらしい(通話料金はほんのわずかだった)。まあ，それはいい。でも，公衆電話だと料金が高くなるっていうのは，どうも納得できない。関係ないけどソニー製のPHS端末を見て，便器を思い浮かべてしまうのは私だけだろうか。(高)

▶新製品の出ないテレコンワールドがつまらない。それでも，番組を見るたび「パンを押し潰すように切るなあ！」「ホースに巻かれて大変だねえ」「ボンネットでハンバーグを焼くんじゃねえ！」「本当に「どうかなあ」なんていってるのか？」などなど，同じ場所で同じツッコミをしてしまう。これはこれで楽しいんだけどね。　(J)

▶ふと見かけた警視庁の求人広告。鑑識要員で化学系……応募者っているのかな？　さて，ちょっとお騒がせした付録ディスクのメディアだが，いろいろあたった結果，なんとか5インチFDのセンでいけることになった。ディスクのメーカーがちょっと変わるが，内容はいつもどおりなので(たぶん……)，安心してほしい。　(U)

▶ちょっと別室でFM/Vのサブノートを使っている(富士通のマシンは77AV以来だ)。借り物なので，プレインストールのアプリとかをバックアップしようと思ったが，これがなんとFD80枚にもなる。別のマシンにはWindow95のβ版をインストールしたが，これまた100Mバイトは使う。ディスクの消費量はこの3年でほぼ10倍ってところかな。　(T)

microOdyssey

趣味としてパソコンを使い始め，いつのまにやらゲーム作りに没頭するようになってからすでに13年が経過した。ふと，その13年間にパソコンの使い方がうまくなったかと思い返してみると，実はそれほどでもないことに気づく。「パソコンを有効活用しているか」と問われると「している」と答えることもできるが，客観的に見て「うまい使い方をしているか」と問われると返答に困ってしまう。

もちろん，仕事のうえで最適な手順でデータを作ったりできることは重要かもしれない。しかし，自由度の高いツールとしてのコンピュータをどう使おうと個人の勝手であり，パソコンを楽しむうえで，うまい使い方なんてものは存在しないだろう。ある程度パソコンを使い込んでくると，使い方に疑問を抱くこともあるだろうが，そんなことで悩むのは時間の無駄。自分なりに効率よく，有効に使う方法を見つけ出し，やりたいことをやるほうがよほど大切といえる。なにかひとつでもいいからやりたいことを見つけられれば，パソコンを楽しめるものだし，目的を達成したときには立派にパソコンを楽しんでいるはずだ。

僕自身，プログラミングができるようになったのも，ゲームを作りたいという明確な欲求があったからだ。その代わり，操作環境にはそれほどこだわらない。これは，X68030を購入したときに，旧マシンのハードディスクを丸ごとコピーするだけだったことからもよくわかる。とにかく，ゲーム作りの環境と仕事のための文書整理ができればこと足りるので，いままでの環境でも問題はない。必要なときに必要なことを覚えるだけで，比較的楽しいパソコンライフを送っている。

いまさら勘違いしている人もいないと思うが，もともとコンピュータがなにかをしてくれるのではなく，コンピュータを使ってなにかをするのだ。目的があいまいだと，いつまでたってもパソコンを楽しんでいるという実感がわかない。

あと「初心者向けの記事を」というアンケートハガキが見られるが，別にOh!Xのとんがった記事を理解できないとパソコンを楽しめないわけではない。もしも必要な情報であると判断したら，理解することもパソコンを使うための楽しみとして捉え，自分なりに調べる努力をしてみるといいだろう。あくまで，自分でやろうとする意志がないかぎり身につくことはないからだ。そのうえで，どうにもならなかったらアンケートハガキを利用すればいい。

メッセージは必ず編集者の目に止まるので，アンケートハガキによる読み手の反響は，かなりの影響力があることを覚えていてもらいたい。問題があったときも読者からの指摘がなければ，編集サイドが勘違いしたまま，問題がそのままに放置される可能性だってありうるのだ。

突っ走り続けるOh!Xではあるが，パソコンを楽しみたいという気持ちは，ごく普通の読者となんら変わりはない。これからも，パソコンを楽しむ道を探していく基本姿勢を変えず，僕は，読者と一緒に最後までOh!Xを作り続けたいと思っている。読者の皆さんも，Oh!Xとのコミュニケーションの手段であるアンケートハガキを使って，積極的に参加しようではないか。　(J)

1995年8月号7月18日(火)発売

特別企画　暑中見舞いPRO-68K

・SX-WINDOW用ファイル管理ツール　DIV.X
・フォント書き換えツール　美麗12ドット.R
・Z-MUSIC ver.2.06/EX-System体験版　ほか

新製品紹介　Zipドライブ

試用レポート　高速SCSIボード(満開製作所)

特別付録　5″2HDディスク　　予価900円

バックナンバー常備店

地域	地区	店名	電話
東京	神保町	三省堂神田本店5F	03(3233)3312
	〃	書泉ブックマートB1	03(3294)0011
	〃	書泉グランデ5F	03(3295)0011
	秋葉原	T-ZONE 7Fブックゾーン	03(3257)2660
	八重洲	八重洲ブックセンター3F	03(3281)1811
	新宿	紀伊国屋書店本店	03(3354)0131
	高田馬場	未来堂書店	03(3209)0656
	渋谷	大盛堂書店	03(3463)0511
	池袋	旭屋書店池袋店	03(3986)0311
	八王子	くまざわ書店八王子本店	0426(25)1201
神奈川	厚木	有隣堂厚木店	0462(23)4111
	平塚	文教堂四の宮店	0463(54)2880
千葉	柏	新星堂カルチェ5	0471(64)8551
	船橋	リブロ船橋店	0474(25)0111
	〃	芳林堂書店津田沼店	0474(78)3737
	千葉	多田屋千葉セントラルプラザ店	043(224)1333
埼玉	川越	黒田書店	0492(25)3138
	川口	岩渕書店	0482(52)2190
茨城	水戸	川又書店駅前店	0292(31)0102
大阪	北区	旭屋書店本店	06(313)1191
	都島区	駸々堂京橋店	06(353)2413
京都	中京区	オーム社書店	075(221)0280
愛知	名古屋	三省堂名古屋店	052(562)0077
	〃	パソコンΣ上前津店	052(251)8334
	刈谷	三洋堂書店刈谷店	0566(24)1134
長野	飯田	平安堂飯田店	0265(24)4545
北海道	室蘭	室蘭工業大学生協	0143(44)6060

定期購読のお知らせ

Oh!Xの定期購読をご希望の方は綴じ込みの振替用紙の「申込書」欄にある「新規」「継続」のいずれかに○をつけ，必要事項を明記のうえ，郵便局で購読料をお振り込みください。その際渡される半券は領収書になっていますので，大切に保管してください。なお，すでに定期購読をご利用の方には期限終了の少し前にご通知いたします。継続希望の方は，上記と同じ要領でお申し込みください。

基本的に，定期購読に関することは販売局で一括して行っています。住所変更など問題が生じた場合は，Oh!X編集部ではなくソフトバンク販売局へお問い合わせください。

海外送付ご希望の方へ

本誌の海外発送代理店，日本IPS(株)にお申し込みください。なお，購読料金は郵送方法，地域によって異なりますので，下記宛必ずお問い合わせください。

日本IPS株式会社
〒101 東京都千代田区飯田橋3-11-6
☎03(3238)0700

Oh!X　7月号
■1995年7月1日発行　定価680円(本体660円)
■発行人　橋本五郎
■編集人　稲葉俊夫
■発売元　ソフトバンク株式会社
■出版事業部　〒103 東京都中央区日本橋浜町3-42-3
Oh!X編集部　☎03(5642)8122
販売局　☎03(5642)8100 FAX 03(5641)3424
広告局　☎03(5642)8111
■印　刷　凸版印刷株式会社

満開の電子ちゃん

作・え 岡村祭（おかむら まつり）

85号（5/18発送）には、酔っちゃう3D迷路「迷ze」とか、SXSIをXellent30で使うとか、数式処理特集とか、「遥か、カナダより」も帰ってきた!!

購読方法：定期購読、ソフトベンダーTAKERU、NIFTY-SERVEでお買い求めいただけます。
また、JCB、VISAカードもご利用になれます（金額9,000円以上の場合）。
★定期購読（送料サービス、消費税込）3ヶ月＝4,500円、6ヶ月＝9,000円、12ヶ月＝18,000円。
・現金書留：〒171 東京都豊島区長崎1-28-23 Muse西池袋2F （株）満開製作所
・郵便振替：02810-6-13298 口座名 電脳倶楽部
・JCB・VISAカード：フリーダイヤル0120-887780 または、NIFTY-SERVE GO MANKAI。
ご注文の際には、郵便番号、住所、氏名、電話番号、タイプ（5インチ・3.5インチ）、新規購読か継続購読かを必ずお知らせ下さい。新規購読の際、購読開始号のご指定のない場合は既刊の最新号よりお送りいたします。製品の性格上返品には応じられませんが、お申し出があれば定期購読を解約し残金をお返しいたします。
★TAKERUでお求めの場合、75号までは1,200円（税込）、76号以降1部1,600円（税込）です。
★お問合わせ先 TEL03-3554-9282（月〜金 午前11時〜午後6時）。
★バックナンバーは創刊号よりございます。★フリーダイヤルは、午前10時〜午後5時。

最近出た別冊の広告がありませんでしたので、まとめていっちゃいますです。1部2千円で〜す。
［七號］地図（北海道南西部・北東北）SuperVoice／FaxioN大魔王／SxGOLF等、音楽オリジナル2曲。
［八號］地図（北海道東北部・終了）メタボールツール／郵便番号最新データ／『じゃんけん将棋』GAWK／AVIPLAY／CDDRV.SYS／VISON／SXZC／『ムーンライト伝説（ZMS）』
［九號］鉄道PDDプロジェクト『THE BASKET2』GCC2 Charlie版（G++とコンパイラドライバ）Xellent30用ツール集『Maybe Tomorrow』全號、他にもツール・画像で一杯！

梅雨に入り、超激安の声いずこ、ジャストのX68kペリフェラル

さて、人生設計をも左右しかねない5月病、皆様は無事に乗り切れたでしょうか？これからは「6月6日、雨がザーザー…」の季節ですよ（6日じゃまだ入梅していませんね、きっと）。下手に設計の良いワンルームマンションにお住まいの方、高い湿度にはくれぐれも注意しましょう。コンピュータの後ろ側で結露した水分がボディに襲いかかっているかもしれませんょ。気がついたらフレーム錆びてたとか、基板に電食の跡がくっきり等、といった状況も考えられます。人間が多少腐ったモノ食べてお腹壊しても簡単に直るでしょうが、機械は自力で直ることはまず考えられません。自分の体より大切にすべきでしょう（笑)。いやー、正直言って錆びてるAlpha XP見たときはビビりましたけどね。え、広告?、まだでしたっけ（笑)。

▽拡張SIMMメモリーボード ER10S

型番：ER10S0n(SIMM未実装）定価￥14,800;ER10SDn(4MByte SIMM1枚実装済）定価￥39,800　対応機種：X680x0全機種（定価はすべて税別）

□クロックスピード10MHzのX68000、今となっては決して速い処理速度とは言えなくなりました。□68000の10MHzもさることながら、このクロックスピードに合わせたメモリー周辺の設計も足を引っ張る要因となっています。これではMPUのクロックを上げてもその効果が充分に生かされないこととなってしまいます。□H.A.R.P.の設計段階で判明していたMPUの高速化に伴うバス等でのウェイトタイムの増大。この無駄な時間をより有効に活用するためのアーキテクチャーがER10の顔です。H.A.R.P.側から見た場合、MPU内部の倍速化された演算処理はストレートにバスに反映されるものの、メモリーアクセスに際して　既存クロックのサイクルで動作するバスのタイミングにあわせた動作をしなければならず、結果として常にウエイトが入っているような状態となります。□ここでER10をバスに接続した場合、バス側で4クロックをワンサイクルとするメモリーアクセスに対し、倍速動作のMPUクロックのアドバンテージを生かし、バス側で1クロック短縮した形でアクセスを完了できるようにタイミングを取る設計としています。□さらに、高速タイプの入手が容易な72ピンタイプのSIMMを採用、さらに内部で使用するゲートICなども高速のものを採用し、全体的な信頼性と安全性の向上に努めています。□毎回同じような解説で申し訳ありませんが、ちょっとしたアイデアでパフォーマンスの向上を図っているという話です。みなさんひとつごひいきに。

▽MPUアクセラレーター H.A.R.P for MC68000

型番：DCMA00D1 定価29,800　対応機種：X68000初代,ACE,EXPERT,SUPER

本体の蓋を外す。シールド板を外す。マザーボード上の68000を外す。□外したMPUの代わりに装着するのはH.A.R.P for MC68000、この後は逆の手順で蓋まで取り付ければ完成、嵐を呼ぶアクセラレーターがその本領を発揮しますょ。マザーボード上のMPUと差し替えるだけの簡単なインストレーション、ハイリスク・ハイリターン（笑）も結構ですが、余計な配線、ハンダ付けもいらないH.A.R.Pの手軽さも「買い」マークですょ。□手軽に倍速化、さらにER10と組み合わせることによってパフォーマンスはさらに向上、組み合わせて使っていただけると「ぐー」ですね。□嵐を呼ぶM68系アクセラレーター、ライト＆エコノミーのH.A.R.Pファミリー、よろしくです。

▽MPUアクセラレーター H.A.R.P-FX (H.A.R.P for MC68030)

型番：DCMA30F1 予価￥54,000　対応機種：X68030をはじめ、MC68030(PGAソケット)が採用されたコンピュータシステム（供給クロック25MHz以下）

□X68030をはじめPGAパッケージタイプ68030を採用するパーソナルコンピューター、ワークステーションのほとんどに適応可能なMC68030互換MPU アクセラレーターH.A.R.P-FXです。X68030への実装時には25MHzのクロックを2倍、オンボード上のMC68030RC50へフルスペック50MHzクロックを供給し、さらにMPUオンチップのキャッシュメモリーがクロックスピードと相乗し優れたパフォーマンスを発揮してくれます。もちろん、ソフトウェアの互換性を完全に維持、既存の環境で動作していたソフトウェアならまず問題なく実行可能でしょう。PRePマシン登場前夜の混沌とした市場をよそに、ひたすら我が道を突き進むH.A.R.P-FX。人呼んで…何度もやると恥ずかしいですね。はぃ。

▽拡張I/Oスロット ESX68

型番：ESX68L4　予価￥39,800　対応機種：X680x0　全機種

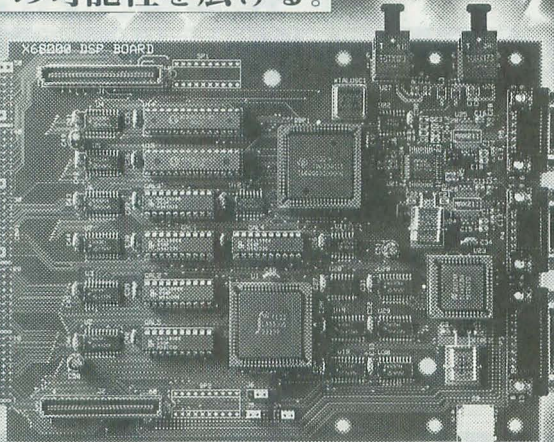
OS-9をはじめ、実はFA系での隠れた需要もあるX680x0、この辺の用途にご利用の皆様には特に拡張I/Oスロットの少なさが問題となっているかと思います。□そんな需要家の皆様、そして純粋にコンピューティングを楽しむユーザーの皆様、外部拡張I/Oスロット はいかがでしょうか？□本体電源に連動する外部スロット専用電源を内臓し、X68k本体とのインターフェースカードは高速タイプのバッファを搭載。加えて3スロット が追加利用できます。□LAN,PIO,GPIB,入れたいカードは何でもどうぞ。□拡張I/OスロットESX68、細く長ーいおつき合いを。

※Motorolaはモトローラ社の登録商標、その他製品の名称等は一般に各メーカーの商標・登録商標です。

サポート
(有)エヌ・エム・アイ

開発・販売
(株)ジャスト

JAST

〒156　東京都世田谷区宮坂3-10-7 YMTビル3F
Phone.03-3706-9766 FAX.03-3706-9761 BBS.03-3706-7134

X680x0を進化させる高速演算DSPプロセッサーボード「AWESOME－X」登場。

この一枚のボードが、X680x0の未来を拓く。高速演算処理によるCGのクォリティアップや制作時間の短縮、128,000bpsのRS-232C高速通信、48kHz高音質デジタルサンプリング、赤外線通信機能などに対応した多機能・高性能化を実現。DSP(Digital Signal Processor)搭載の高速演算プロセッサーボード「AWESOME-X」が、あなたのX680x0を、新たな可能性の世界へと進化させます。

■主な仕様 ●DSP:TEXAS INSTRUMENTS社 TMS320C26B-40MHz ●RAM:DSPワーク64KB, I/F 4KB●RS-232C:D-sub9pin×2●EXT 1:EIAJ準拠 光デジタルオーディオI/F入出力端子●EXT 2:赤外線通信用I/F●EXT 3:拡張I/F ■付属ソフトウェア(予定) ●FLOAT2.X互換FLOATドライバ●DSP直接制御FLOATドライバ●高速シリアルドライバ●シリアルMIDIドライバ●PCMドライバ●JPEGデコーダ／エンコーダ●セルフプログラムチェック●ベンチマークプログラム■オプション(予定) ●MIDIドーターボード(純正MIDIボード互換)●赤外線通信ユニット(赤外線通信、電子手帳とのリンク)●Maximum Over Drive Processorボード(TMS320C3x搭載アクセララレータボード)

標準価格￥89,800（税別）

DSP INJECTION forX680x0
AWESOME-X
X680x0用DSP高速演算プロセッサーボード

GRAVIS

企画・開発/(有) グラビス〒213 神奈川県川崎市高津区坂戸3-2-1 かながわサイエンスパーク東棟513　tel:044(812)7499 FAX:044(813)7243

＊TMS320C26B,TMS3203xは,TEXAS INSTRUMENTS 社の登録商標または商標です。＊X680x0、は,シャープ株式会社の登録商標または商標です。

製品のお問い合わせは、サポートセンター（0286）27-1829までFAXで、またはTECOSY3でどうぞ。

好評発売中

SXパワーアップ委員会 シャーペンワープロパック

標準価格 ¥6,800

SXパワーアップ委員会シリーズ第1弾は、シャーペンをさらに強化する「シャーペンワープロパック」です。

シャーペンワープロパックをインストールすることによって、シャーペンが限りなくワープロに近い存在へとパワーアップします。

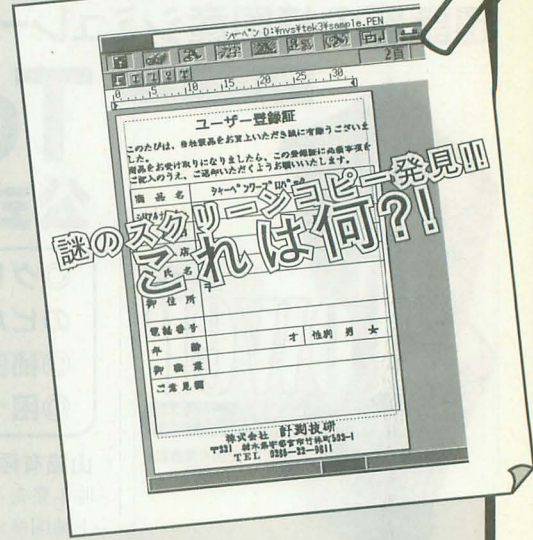
文字の回転や各種タブ、インデントなど、最新ワープロソフトにも負けない表現力を追加するほか、文系ユーザー待望の縦書き表示、縦書きインライン入力もサポート。それでいて、従来通りの軽快さもそのまま継承しています。

●動作環境
・SX-WINDOW Ver3.1以上
・空きメモリ300KB程度

●シャーペンに追加される主な機能
・縦書き入力
・文字の回転
・ルーラ(定規)の表示
・各種タブ(均等割付など)およびインデントの設定*
・各種禁則処理(追い込み均等など)*
・行揃えの拡張*
・段組み印刷

*:パラグラフごとに設定可能

●プログラマ向け機能も充実
・編集中のソースをコンパイルする等、マクロ機能を強化

●付録
・シャーペン外部コマンド開発キット(ライブラリおよびリファレンス)
・IFM ver 4.0

SX-WINDOW用CD-ROM辞書検索ソフト

SX広辞苑《EPWING対応版》

標準価格 ¥19,800　岩波書店「広辞苑第4版」CD-ROM版 バンドルセット ¥43,800

・豊富でパワフルな検索方法により、必要な情報をすばやくピックアップ。
・広辞苑の最新版である第4版をもとにしたCD-ROMを使用するので、よりコンテンポラリーなキーワードにアクセス可能です。
・シャーペンと融合して語句の検索を行なうシャーペン用外部コマンド"LightWing.X"を同梱。複雑な検索を行なう場合はSX広辞苑.Xを、普段よく使う単純な検索にはLightWing.Xを、という使い分けも可能です。
・広辞苑第4版CD-ROM版と同様に、EPWING(V1)規約にもとづいたCD-ROMタイトルなら、ほとんどのCD-ROMの内容を検索できます。

●動作環境
・SX-WINDOW 3.0以上
・SX-WINDOW動作中の空きメモリとして1MB以上を推奨
・CD-ROMドライブ(CD-ROM Driver Ver2.0が付属するので、CD-ROM Driverを別途お買い上げいただく必要はありません。CD-ROM Driverのマニュアルや添付ソフト等は付属しません)

68040搭載アクセラレータ

X68040turbo

標準価格 ¥98,000
ヒートシンク別売 ¥1,000

040turboは、68040を搭載したX68030(5インチタイプ)専用のアクセラレータです。040turboを装着することで得られるパフォーマンスは、従来の2～3倍! 計算、特に浮動小数点演算中心のソフトならば、さらにそれ以上の高速化も望めます。

詳しくはソフトバンク刊「X68040turbo～A Story of Makeing "After X68030"～」(BEEPs著)をご覧ください。

X680x0用Ether net接続パック

EthernetStarterPack/X680x0

標準価格 ¥88,000

ESP/Xは、Ether netアダプタ「Ether+」と、TCP/IPドライバ、そして基本的なアプリケーションからなるパッケージです。

ftp、telnet(いずれもクライアント)等、基本的なアプリケーションを標準添付。ドライバを活用するためのライブラリも付属します。

※10BASE-2対応モデル・10BASE-T対応モデルの2種類があります。

●動作環境
・Human68k ver3.0以上
・メモリ常駐量500KB前後
・SCSIインターフェース内蔵機種以外はSCSIボードが必要

SCSI-2対応CD-ROMドライブ専用ドライバ

CD-ROM Driver Ver2.00

標準価格 ¥4,800

X680x0用フリーソフトウェアCD-ROM

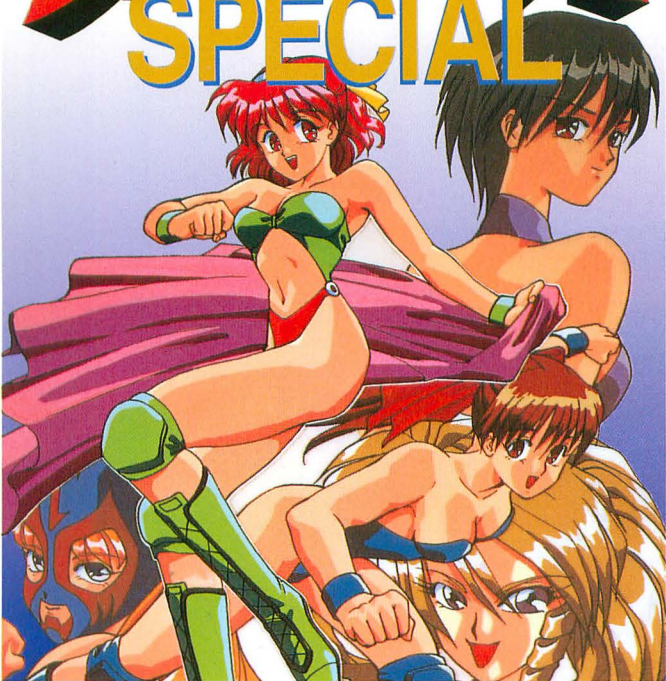
FreeSoftwareSelection Vol.2

標準価格 ¥6,000

お求めはお近くのパソコンショップ、または当社通販部(TEL:0286-22-9811)へお申し込みください。
通販ご希望の方は、ソフト代金+送料¥1,000に消費税を加え、ご住所・お名前・電話番号・商品名を明記した紙を同封の上、現金封筒でお申し込みください。

低金利クレジット　通信販売送料 全国一律¥1,000 長期クレジット可能　※表示価格に消費税は含まれておりません
株式会社 計測技研 マイコンショップ BASIC HOUSE 本社／ショールーム／通販部
〒321 栃木県宇都宮市竹林町503-1
TEL 0286-22-9811　FAX 0286-25-3970
サポートネット TECOSYS-3 24時間稼動中! (0286)51-1430 (9600bps MNP5)

ハイパーアクション プロレスカードゲーム

レッスルエンジェルス SPECIAL

セクシーでパワフルな女子プロを制覇しろ！

18禁版

カードバトルにプロレスを融合させた、「レッスルエンジェルス」シリーズ。いよいよ最大のヒット作「レッスルエンジェルススペシャル」が登場です。さまざまなイベントの選択によって運命が変わる、マルチシナリオ・マルチエンディング。プロレス技数、カテゴリーが増加して、レスラーの個性もパワーアップ。そして、「恐怖の水着はぎデスマッチ」もパワーアップして復活！18禁だから、そのセクシー度はもうケタ違い！待望のX68000移植完成！明日のトップイベンターを目指すのだ！

機能アップ！

- ●オリジナルオープニングを収録
- ●画面のレイアウトを変更
- ●エキジビションモードグラフィック描き直し
- ●256色モードと16色モードを搭載
- ●サウンドも明るめに変更
- ●AD-PCMによる効果音
- ●ディスクアクセスを最少に抑える設計

このソフトは、全国のパソコンショップで、パッケージ版で販売いたします。TAKERUでは販売致しません。TAKERU事務局では通信販売はいたしませんので、悪しからずご了承下さい。

対応機種：X68000/X68030
要メモリ2Mバイト
（ハードディスク対応）

制作：グレイト

￥8,800（税別）

三國志

知力の極限に挑む、君主、武将、軍師の膨大なデータ。小説よりリアルと、名作の誉れ高い中国統一ゲーム。この歴史的な傑作シリーズはどのようにして始まったのか?SLGファンなら絶対に見逃せない!!

制作/光栄
対応機種/X68000（30不可）　**¥5,200**

太閤立志伝

裸一貫の足軽頭から身を興し、関白にまで登り詰めた男・木下藤吉郎(豊臣秀吉)。草履を温めたエピソード・奇跡の墨俣一夜城など、数々の逸話を持つ男の一生を再現する、リコエイションゲームの傑作です。

制作/光栄
対応機種/X68000（30不可）　**¥3,400**

ファランクス

デカキャラ・派手め演出の横スクロールオアワーシューティング。拡大・回転・縮小・多関節・半透明・ラスタースクロール・MIDIと、各種要素がいっぱい詰まってます。

制作／ズーム
対応機種／X68000（30不可）　**¥2,500**

三國志 II

登場人物350余名、最大11人まで同時プレイ可能、6編のマルチシナリオ方式、埋状の毒・駆虎呑狼等のユニークな計略要素導入、さらに深みを増した外交・HEX戦など、まさに名作!カシオペアの向谷　実のBGMも話題に。

制作/光栄
対応機種/X68000（30不可）　**¥4,900**

蒼き狼と白き牝鹿　元朝秘史

光栄歴史三部作の一角を成す、草原の英雄チンギス・ハーン。稀代のスケールと空前絶後の迫力で、一代帝国を築き上げた男の豪快な一生を見事に再現!熱いシミュレーションの傑作です。

制作/光栄
対応機種/X68000（30不可）　**¥3,400**

A列車で行こうII

かの「A列車」シリーズの第2弾。パズル的要素がアツクなる!鉄道会社社長の立場で、線路の敷設・撤去を行い、ワールドワイドにマップを発展させていこう。

制作／アートディンク
対応機種／X68000（30不可）　**¥3,800**

大航海時代

リコエイションゲームシリーズの傑作。毎回違った展開が楽しめるイベントジェネレーティングシステム。帆船の特徴が活かされたHEX戦。失われたロマンを求めて、冒険者たちの航海の旅が始まる。

制作/光栄
対応機種/X68000（30可）　**¥3,400**

ロイヤルブラッド

新シリーズ「イマジネイションゲーム」のデビュー作。イシュメリアという架空の島国を舞台にした、幻想世界のシミュレーションゲームだ。あなたは独立貴族のひとりとなり、領主達が持っている6つの宝石を集め、イシュメリアの新王となれ!

制作/光栄
対応機種/X68000（30可）　**¥2,700**

AIII（A列車で行こう3）

さらにワイドに、さらに完成度の増した、世界レベルヒットの第3弾。世にA.IIIブームを巻き起こしたことで、記憶に新しい超有名作、ついに文庫に登場！

制作／アートディンク
対応機種／X68000（30可）　**¥3,800**

維新の嵐

坂本龍馬が、西郷隆盛が、吉田松蔭が日本を憂い、改革を目指して奮い立つ!幕末の志士の個性を際だたせる緻密なパラメータ。出会いの楽しさ、駆け引きを楽しむ新システム。強力な機能で、維新を操れ!

制作/光栄
対応機種/X68000（30不可）　**¥3,400**

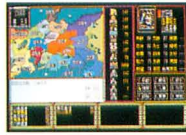
ヨーロッパ戦線

戦乱のヨーロッパ。砂塵の彼方から迫り来る黒い車体は、敵か味方か?次々に飛び込んでくる情報、時事刻々と変わる戦局。多彩な兵器やユニット、人間的要素を重視した各種パラメータ。WWIIシリーズ第2弾。勝利の旗を手に入れろ!

制作/光栄
対応機種/X68000（30可）　**¥4,500**

栄冠は君に

高校野球シミュレーションシリーズの、記念すべき第1作。全国制覇を達成するには、3990校の頂点に立たなければならない。感動の優勝セレモニーを、果たして見ることが出来るか！？

制作／アートディンク
対応機種／X68000　**¥3,800**

信長の野望　戦国群雄伝

400余名の群雄が割拠する下剋上の乱世。配下の羽柴秀吉、柴田勝家を個性豊かな武将たちを、思いのままに操って、戦雲たなびく戦場へ、天下分け目の決戦に臨む！光栄の代表作「信長の野望」シリーズの傑作！

制作/光栄
対応機種/X68000（30可）　**¥3,400**

大戦略 III '90

90年代にふさわしくパワーアップされた「大戦略」シリーズ。戦略思考ルーチン、ゲームスピード、コマンド体系、リアルタイムオペレーションなど大幅革新された作品です。

制作／システムソフト
対応機種／ X68000　**¥2,500**

ルーンワース「黒衣の貴公子」

ハイドライドシリーズに続く、新ARPGシリーズ第1弾。綿密に構築された世界「ルーンワース」を舞台に、極めて自由度の高いゲームシステムの中で、興奮の冒険が始まります。

制作／T&Eソフト
対応機種／X68000　**￥700**

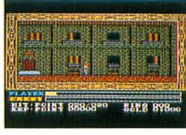
伊忍道　打倒信長

1つのゲームでSLGとRPG、2つのジャンルが楽しめるリコエイションゲームの第3弾。特にRPGの要素が濃い、異色傑作だ!意志を持ったキャラクターが目的に向かって行動を展開。敵を倒して腕を上げ、技を磨いて信長を倒せ!

制作/光栄
対応機種/X68000（30不可）　**¥3,400**

ジェノサイド 2

あのズームのゲームがついに名作文庫に登場！特大キャラとハデハデな演出で、68ユーザーのどぎもを抜いた名作アクションゲームだ。MIDIにも対応しているぞ。

制作/ズーム
対応機種/X68000（30不可）　**¥2,500**

イースIII（ワンダラーズフロムイース）

よりアクション性を増した、これまた、大人気を博したアクション・ロールプレイング。アドルの最後の冒険物語でした。攻撃方法もいっそう多彩になって、時間を感じさせない逸品です。

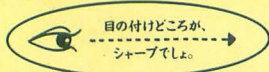
制作／日本ファルコム
対応機種／X68000（30不可）　**¥2,000**

パソコンソフト
自動販売機
TAKERU

TAKERU事務局
〒467　名古屋市瑞穂区苗代町2番1号
ブラザー技術開発センタービル2F
TEL(052)824-2493（受付時間：月～金　13:00～18:00）

営業所
東京営業所
(03) 5443-4967
大阪営業所
(06) 258-3024

通信販売 1994年4月1日より、送料／手数料が有料になりました。

ソフト名、機種名、メディアのサイズ、住所、氏名、電話番号を明記の上TAKERU事務局まで現金書留でお申し込みください。送料／手数料は、1回のお申し込み総金額が5,000円以上の方は無料。4,900円までの方は500円をいただきます。4,900円までの方は現金500円をプラスしてお申し込みください。誠に勝手ながら、皆様のご理解とご協力の程、お願い申し上げます。

感性を光らせる。

さまざまなフィールドで、研ぎ澄まされた感性に応える潜在能力の実証

X68の潜在能力は、まさに時代とともに証明されつつあります。
開発当初より、現在のマルチメディア環境を想定していた事実。
グラフィック能力はもちろん、ADPCM対応、オリジナルウィンドウシステム、
X68にとってこれらは、数年前のスペックなのです。
パソコンの存在そのものを革新した「創造性」、マインドを喚起する「こだわり」、
いま、先見のユーザーに支えられたX68は
そのコンセプトの開花を得て、多彩なフィールドへと飛翔します。

Workbench
WSとしての楽しみ

たとえば、リアルタイム・マルチタスク・
オペレーティング・システムOS/9。
X68030の能力を最大限に引き出す
UNIXライクな操作性と洗練された機能。
X-WINDOWや動画ツールのサポートで
さらに深い楽しみが…。

※OS/9はマイクロウェア・システムズ㈱の登録商標です。
※UNIXは、X/Openカンパニーリミテッドが独占的にライセンスする米国および他の国における登録商標です。

Create
創造するよろこび

SX-WINDOW開発支援ツールが
創造力を刺激する。
ソフト開発に必要なツールや
サンプルプログラムを多彩にバンドル、
ウィンドウ上で効率よく作業でき、
初めてプログラムに挑む人への
やさしい配慮が、創造するよろこびを
さらに高めてくれるでしょう。

Ammusement
遊びへのこだわり

X68の能力の高さを端的に示す
アミューズメントフィールド。
マインドをきわめたゲームフリークの
熱い期待に応える。
画像の美しさが感性を刺激する、
さらにパワーアップされた
「スーパーストリートファイターII」なら、
キミのこだわり度は今、全開！
© CAPCOM ALL RIGHTS RESERVED

X68030 32bit PERSONAL WORKSTATION / **X68000** PERSONAL WORKSTATION・XVI

X68030 [本体+キーボード+マウス・トラックボール]
130㎜FD(5.25型)タイプ CZ-500C-B(チタンブラック) 標準価格398,000円(税別)・〈HD内蔵〉CZ-510C-B(チタンブラック) 標準価格488,000円(税別)
X68030 Compact [本体+キーボード+マウス]
90㎜FD(3.5型)タイプ CZ-300C-B(チタンブラック) 標準価格388,000円(税別)
X68000 XVI Compact [本体+キーボード+マウス]
90㎜FD(3.5型)タイプ CZ-674C-H(グレー) ＊

●ディスプレイは別売です。●消費税及び配送・設置・付帯工事費、使用済み商品の引き取り費等は、標準価格には含まれておりません。●画面はハメコミ合成です。
＊〈標準価格〉表示のない商品の価格については、販売店にお問い合わせください。

■お問い合わせは… シャープ株式会社 電子機器事業本部システム機器営業部 〒545 大阪市阿倍野区長池町22番22号 ☎(06)621-1221(大代表)

7月号 1995年7月1日発行（毎月1回1日発行） 通巻159号 昭和58年11月2日第三種郵便物認可 ソフトバンク株式会社発行 Printed in Japan 定価680円（本体660円）

T1002179070685 雑誌 02179-7